ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

DESARROLLO DE UN SISTEMA ROBOTIZADO PARA INSPECCIÓN VISUAL DE PIEZAS

Memoria

TRABAJO FINAL DE GRADO

AUTOR:

DAVID MARÍN MARTÍNEZ

TUTOR:

ANTONIO JOSÉ SÁNCHEZ SALMERÓN

RESUMEN

El objetivo de este trabajo de fin de grado ha sido el desarrollo de un sistema robótico apoyado en un sistema de visión artificial.

Para realizarlo tomamos como base un robot IRB 140 de la marca comercial ABB y se le acompaña de un sistema de visión artificial compuesto por la cámara JAI CV-M77 y gestionada por el programa de visión “Sherlock”. El sistema de visión indicará al robot en qué posición debe coger las piezas y en qué lugar debe dejarlas.

Índice

1. Breve historia de la robótica y la visión artificial

1.1 Historia e inicios de la robótica

1.1.1 Inicios

Muchas personas, cuando oyen hablar de robots o de robótica, piensan en unos aparatos realmente complicados y, sobre todo, que se usan desde hace relativamente poco tiempo.

¿Es correcto? La verdad es que el robot tal y como lo conocemos ahora, sí que es un concepto bastante moderno. Sin embargo, el interés del ser humano por copiar a la naturaleza y por crear herramientas y artefactos que hagan más fácil su día a día en su hogar y en el trabajo hizo que, hace ya muchos siglos, se empezara el camino que acabaría desembocando en el campo de la robótica.

Sus inicios se remontan al siglo I d.C., cuando Herón de Alejandría creó los primero autómatas, incluida la primera máquina de vapor.

Siglos más tarde, alrededor de 1495, el propio Leonardo da Vinci creó un robot humanoide, el cual era capaz de sentarse, mover los brazos, el cuello y la mandíbula.

En el siglo XIX, en Japón, un artesano creó diferentes mecanismos que servían té, disparaban flechas o pintaban [1].

1.1.2 Crecimiento

Y en los años 1950 se instala el que es el primer robot industrial, el robot “Ultimate”, creado por George Devol, en una planta de General Motors. La máquina realizaba el trabajo de transportar las piezas fundidas en molde hasta la cadena de montaje y soldar estas partes sobre el chasis del vehículo, una peligrosa tarea para los trabajadores, quienes podían exponerse a inhalar los gases de combustión de la soladura o a perder un miembro si no llevaban precaución.

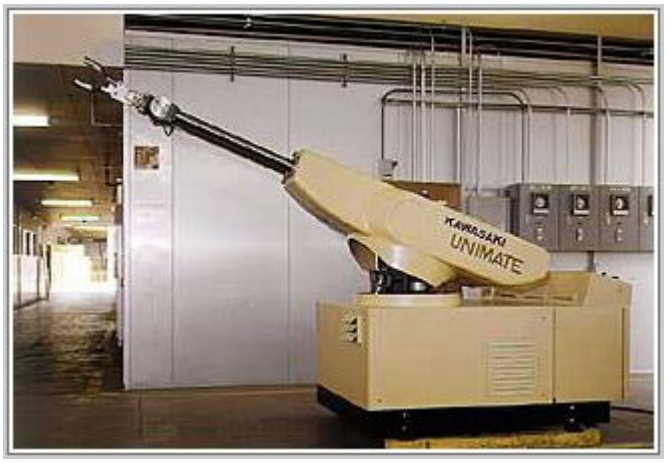


Ilustración 1. Robot Ultimate.

1.1.3 Consolidación

Más tarde, en 1975, el ingeniero Victor Scheinman, desarolló un manipulador flexible que era capaz de mover un objeto y colocarlo en cualquier orientación. Se le llamó P.U.M.A. (Brazo Articulado Universal Programable por sus siglas en inglés) y es la base de los robots industriales de la actualidad [2].

Ilustración 2. Robot P.U.M.A.

Actualmente existen fábricas flexibles están totalmente automatizadas. Algunos trabajos de Investigación publicados recientemente son [3] y [4].

1.1.4 Polémica

Dado que las grandes empresas han implementado en sus líneas de fabricación y montaje unidades robóticas que antes desempeñaban trabajadores remunerados, surge la pregunta: ¿la robótica elimina puestos de trabajo?

En primer lugar pensemos qué tareas asumen los robots: ensamblaje, soldaduras, manipulación de cargas pesadas, pinturas en ambientes tóxicos, tareas repetitivas y pesadas. Por ejemplo, hace años en las fábricas de coches, las líneas que soldaban las carrocerías del coche se basaban en dos trincheras repletas de operarios con pinzas neumáticas mientras el pasaba por mitad de ellas. Los operarios estaban expuestos a las explosiones propias de las proyecciones por soldadura, al calor y a los esfuerzos de mover manualmente las pinzas de soldadura. En estos momentos en las plantas de carrocerías son los robots los que sueldan y los operarios los alimentan con piezas y reparan posibles averías. Como vemos, la calidad del trabajo de los operarios mejora significativamente.

De ese modo podemos afirmar que la robótica no destruye empleos, sino que modifica la oferta laboral, demandando más personas con capacidades y conocimientos técnicos para el montaje, desarrollo, implementación y mantenimiento de las líneas de producción. Y para muestra un botón: Japón (con más de 300.000 robots), Estados Unidos (24.000) y Alemania (18.000) tienen tasas de desempleo mucho menores que los países sin tanto parqué robótico.

1.2 Historia e inicio de la visión artificial

1.2.1 Inicios de la visión artificial.

Mediante una cámara obtenemos imágenes planas de un mundo tridimensional. Pero el interés en representar el mundo tridimensional en dos dimensiones existía desde tiempo antiguo, como es el caso de la pintura.

Las propiedades geométricas de la proyección eran conocidas ya en tiempos de la Grecia clásica. Thales de Mileto y Euclides ya hicieron avances utilizando conocimientos de geometría.

En el renacimiento tuvieron lugar muchos grandes avances. En este caso los pintores italianos, como Bruneleschi, estudiaban las imágenes con el fin de reproducir correctamente los efectos de la perspectiva, en contraposición con lo que hacían las pinturas de épocas anteriores, que no mostraban diferencias de proporciones de los objetos.

Para muestra, véanse las siguientes imágenes:

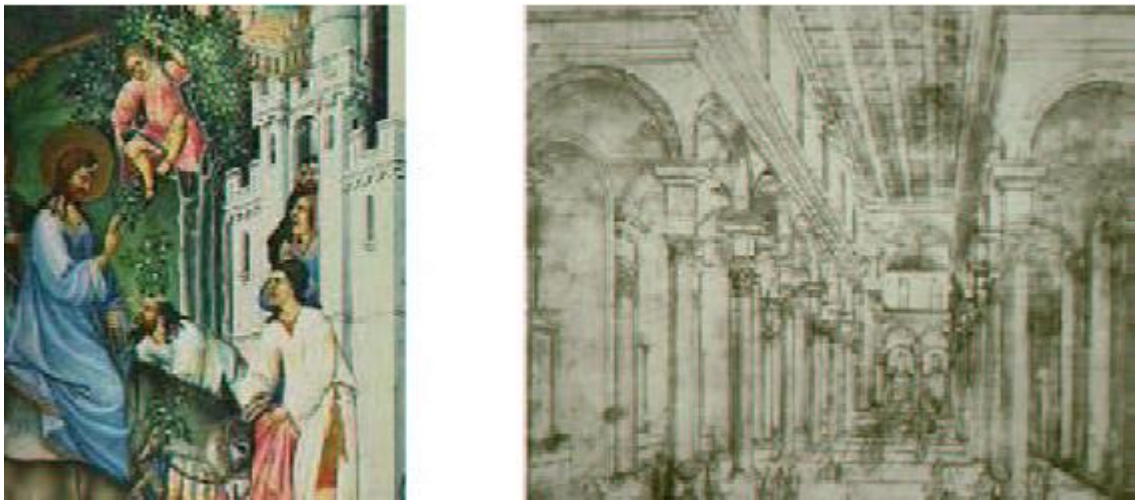


Ilustración 3. Izquierda: pintura pre-renacentista (Jesús entrando a Jerusalén) y a la derecha, pintura renacentista (Iglesia del Espíritu Santo, Bruneleschi).

En 1545 se presenta la cámara oscura, creada por el astrónomo Gemma Frisius (1508-1555). En esta cámara la luz entraba por un orificio muy pequeño y proyectaba la imagen invertida del mundo exterior.

En 1826, utilizando esta cámara oscura y una superficie fotosensible, el químico francés Niepce (1765-18833) toma la primera fotografía.

En la actualidad, se utilizan cámaras réflex y CCD que emplean lentes para incrementar la potencia de la luz y mejorar el enfoque de la imagen [5].

1.2.2 Evolución histórica

El fundamento de lo que hoy conocemos como visión artificial está en el programa espacial de la NASA, en 1964. Este programa utilizaba cámaras digitales que enviaban la información en bits y bytes dese el satélite, mientras volaba hacia el planeta Marte. Fue con el procesado digital de dichas imágenes con lo que se dio inicio a lo que actualmente es el campo de la visión artificial.

1.2.3 Situación actual sobre el desarrollo de la visión artificial

La visión artificial se trata de un campo relativamente nuevo. En sus inicios la computación era, comparada con ahora, muy básica y procesar muchos datos de imagen era realmente complicado. No fue hasta que los procesadores empezaron a adquirir más potencia que el campo de la visión artificial despegó.

Existen abundantes métodos para resolver varias tareas bien definidas en visión artificial [6]. En la mayor parte de las aplicaciones, se usan programas de visión artificial con funciones pre-programadas para resolver tareas en particular [7-9] [10] y [11].

Por otra parte también existen muchas aplicaciones donde se integra la visión con la robótica para dotar de mayor flexibilidad a las líneas de producción [12-15].

2. Objetivos, dudas y problemas.

2.1 Objetivos:

Se nos presenta una cinta transportadora que va a traernos piezas que deberemos ir colocando en sus envases. Las piezas en la cinta pueden ir en cualquier posición, al igual que el envase, que puede estar en cualquier orientación.

2.2 Soluciones:

2.2.1 Solución 1

Como las piezas pueden ir en cualquier posición, diseñar un sistema mecánico que iguale la posición de la todas las piezas en la cinta. Para el envase, mediante centradores, diseñar un sistema en el que el envase sólo se pueda colocar en una posición y orientación predeterminada.

Pros:

- Programación de robot fija.
- Poca inversión.

Contras:

- Imposibilidad de utilizar otras piezas en la misma cinta.
- Imposibilidad de utilizar contenedores diferentes.

2.2.2 Solución 2

Podemos utilizar un sistema de visión artificial, el cual le comunicará al robot en qué posición debe coger las piezas y dejarlas.

Pros:

- Versatilidad: con el mismo programa de robot y modificando el programa de cámara, podemos hacer el diseño para una gran variedad de piezas y de contenedores.
- Eliminamos sistemas mecánicos y sus posibles problemas: atascos, volcado de piezas, deformaciones…

Contras:

- Mayor inversión inicial.

2.3 Solución elegida y justificación

La solución elegida es la número 2. Aunque a priori pueda parecer más compleja, las ventajas inherentes a utilizar la visión artificial en los procesos industriales hacen que nuestra solución sea más abierta y más adaptable a futuros cambios de producción o a líneas en las que se fabrica más de un producto. A la vez la visión artificial ha demostrado ser un sistema estable en el tiempo, es decir, que no existe la necesidad de calibrarlo constantemente. Además, con una correcta programación, obtendremos un sistema muy robusto a prueba de errores.

3. Qué utilizaremos

A continuación vamos a ver las imágenes de los distintos sistemas que utilizaremos para el proyecto:

1. Robot: el robot que utilizaremos es el ABB modelo IRB140. Se trata de un brazo articulado de 6 ejes.

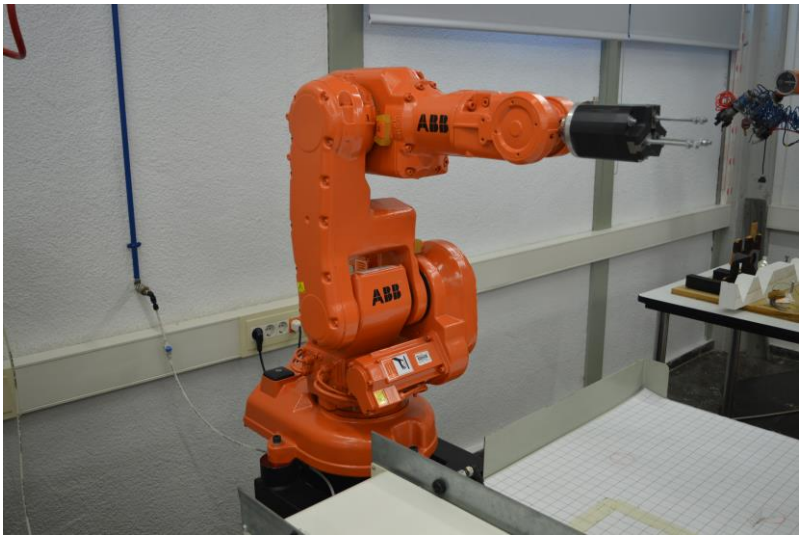


Ilustración 4. Robot ABB IRB 140

2. Armario del robot: en el armario se encuentran los controladores, las entradas y salidas digitales y analógicas, los servos, las diferentes conexiones, como el USB.

Ilustración 5. Unidad de Control

3. Flex pendant: con esta herramienta podemos manejar el robot, cargar, copiar, crear y eliminar programas, activar salidas, ejecutar el programa, pararlo, etc.

Ilustración 6. FlexPendant

4. Cinta transportadora. Con esta cinta transportadora moveremos las piezas hasta la zona de cogida, delimitada por un sensor láser incorporado en la propia cinta.

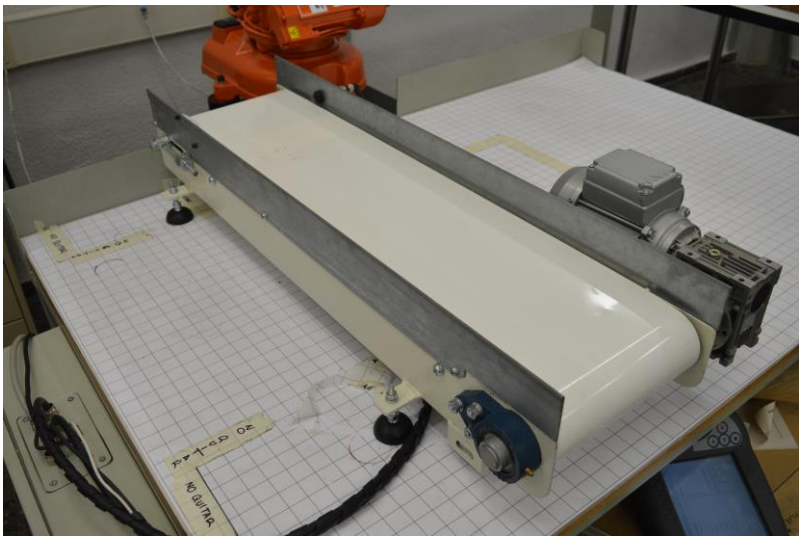


Ilustración 7. Cinta transportadora

5. Armario cinta. Aquí están los elementos eléctricos que controlan la cinta. Deberemos subir el interruptor IF1 para que la cinta funcione.

Ilustración 9. Armario cinta

Ilustración 8. Interior del armario de la cinta

6. Cámara modelo JAI CV-M77.

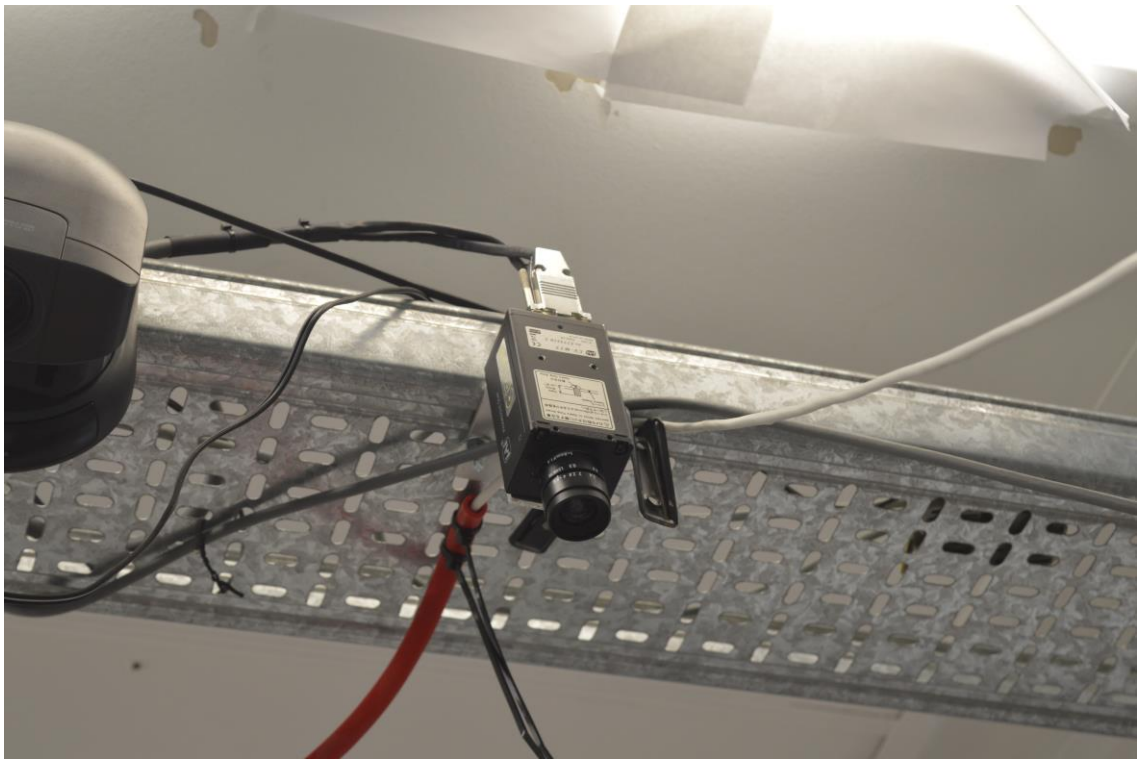


Ilustración 10. Cámara.

6. IPD. Es el computador que controla el sistema de visión artificial.

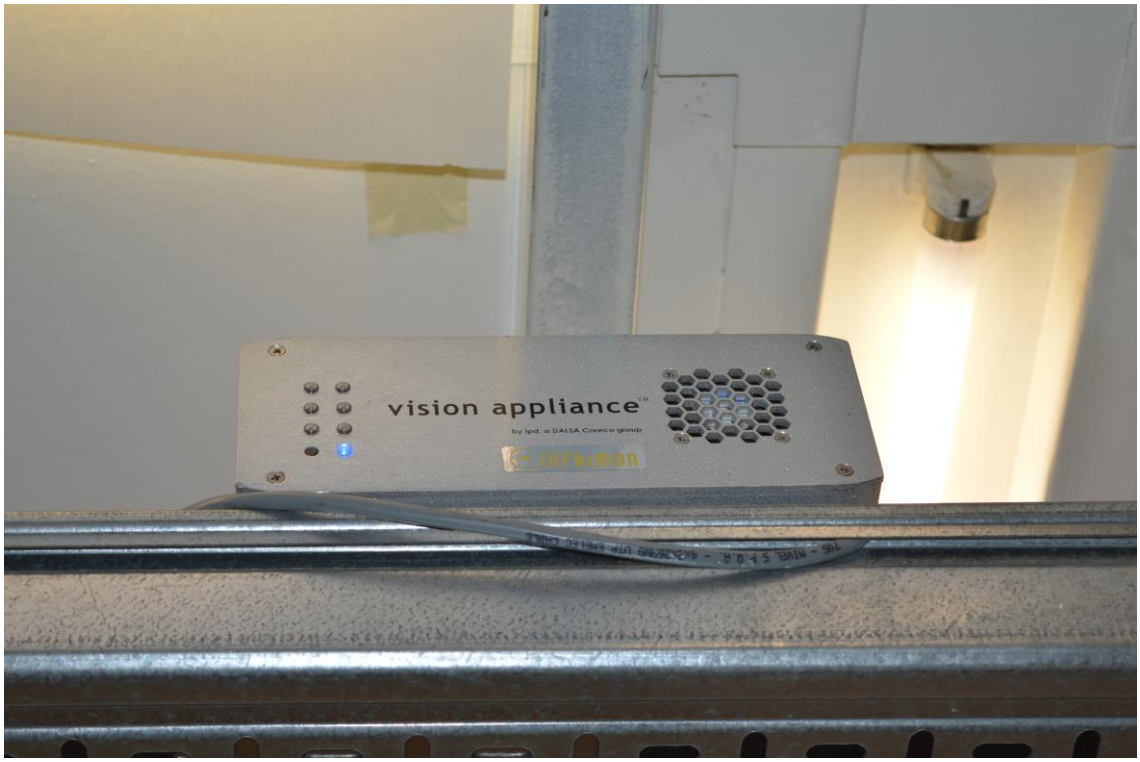


Ilustración 11. IPD

7. Ventosa. Controlada por el robot.

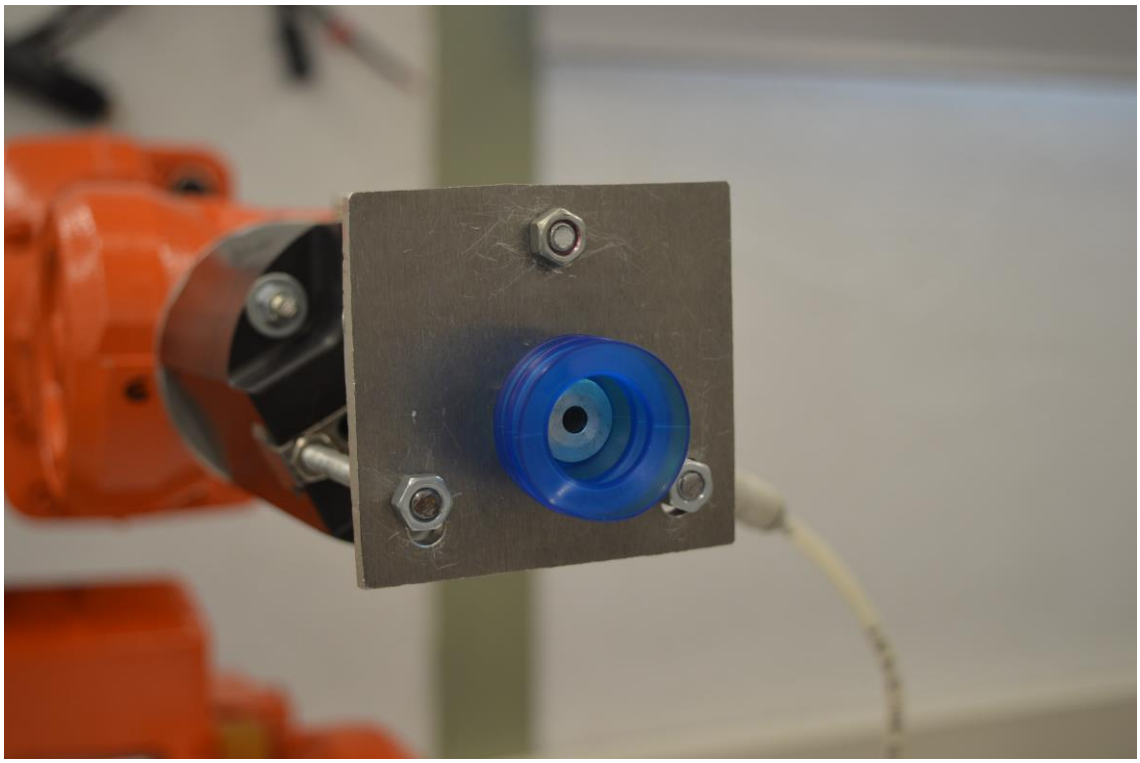


Ilustración 12. Ventosa.

8. Objetos a manipular

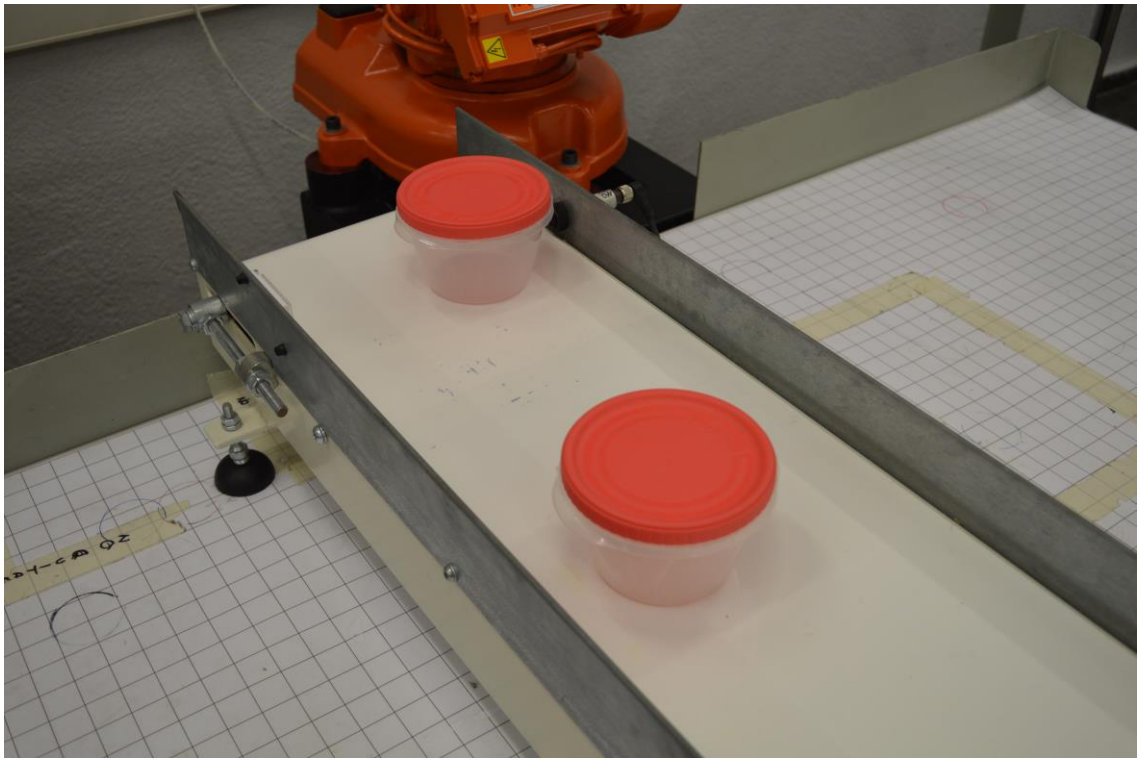


Ilustración 13. Objetos a manipular

4. Arranque de los equipos

Antes de proceder con calibraciones y datos técnicos y de programa, vamos a ver cómo encender tanto el robot como el sistema de visión artificial y cómo crear programas en ellos.

4.1 Arrancar el robot IRB 140

Para arrancar el robot debemos girar el interruptor que se encuentra en la Unidad de control hasta la posición ON. Encima de esta Unidad de Control encontramos los pasos necesarios que debemos seguir tanto para encender como para apagar el robot, siendo necesario seguirlos.

4.2 Crear programa en RobotStudio

El lenguaje de programación que usan los robots de ABB se llama Rapid, y podemos crear estos programas desde una completa aplicación llamada RobotStudio, donde no solo podemos crear el programa sino también simular líneas enteras con múltiples robots, cintas, etc.

Para crear un programa en RobotStudio es necesario realizar los siguientes pasos:

1. Abrimos RobotStudio. A continuación nos aparecerá una ventana desde donde podremos crear nuevas estaciones vacías, estaciones con sistemas de robot y módulos de Rapid. También podemos cargar estaciones ya guardadas. En este caso, vamos a iniciar un nuevo proyecto. Así que pulsaremos en “Estación con controlador de robot” y a continuación, a la derecha, elegiremos el robot con el que estamos trabajando, que en este caso es el IRB140 (no confundir con el IRB140T). Pulsamos crear.

   En la siguiente ilustración podemos ver en detalle cómo hacerlo:

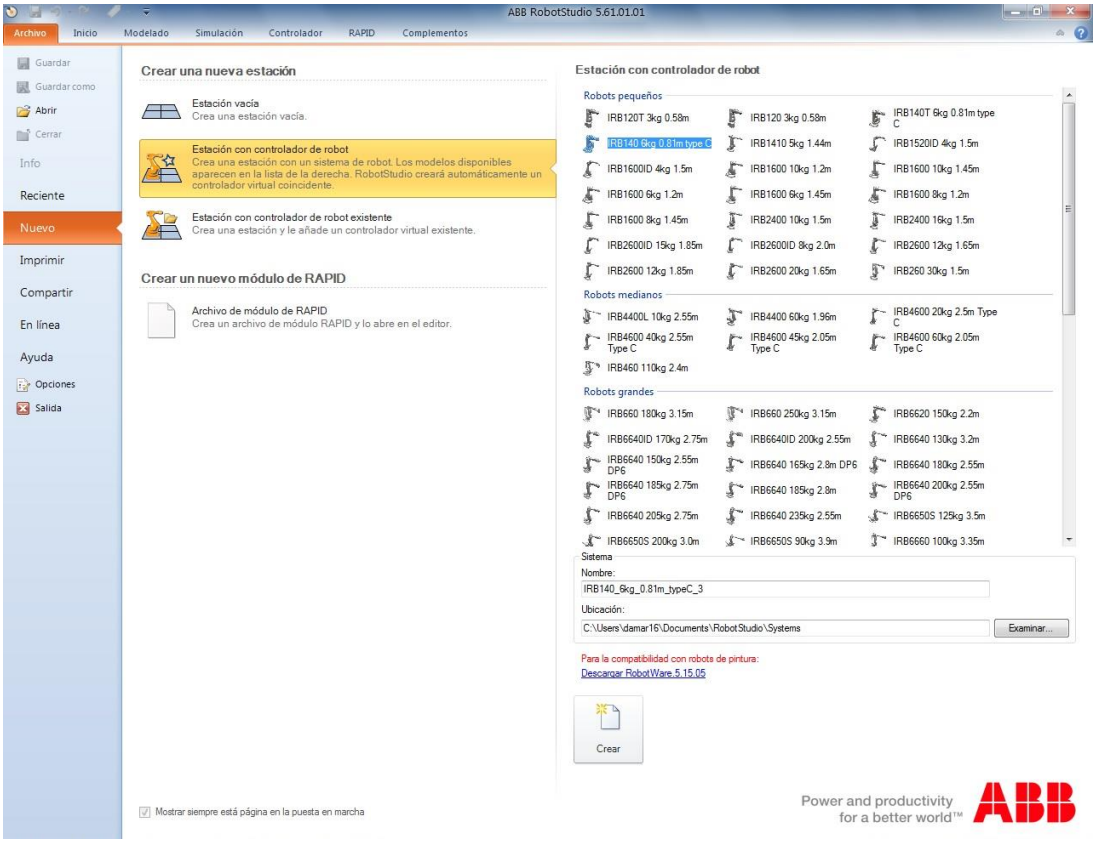


Ilustración 14. Seleccionar controlador estación.

2. A continuación nos aparecerá una ventana donde debemos seleccionar el mismo robot que hemos puesto antes, el IRB140.

Ilustración 15. Seleccionar biblioteca.

3. Seguidamente, vamos a la pestaña “Controlador” y allí dentro del sistema de robot, abrimos los desplegables hasta llegar a los módulos de Rapid. A continuación damos con el botón derecho del ratón sobre T_ROB1 (o el nombre que le hayamos puesto) y pulsamos Nuevo módulo.

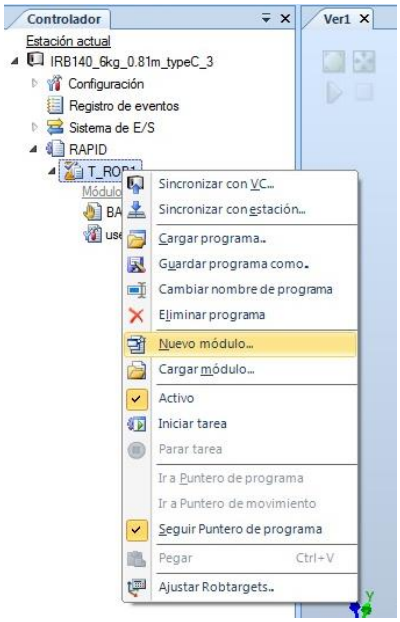


Ilustración 16. Nuevo Módulo de programa.

4. Después de pulsar nos aparece una ventana como la de la siguiente ilustración, donde tendremos que elegir en “Tipo de módulo” la opción programa y en el nombre de módulo pondremos “MainModule”. Aceptamos.

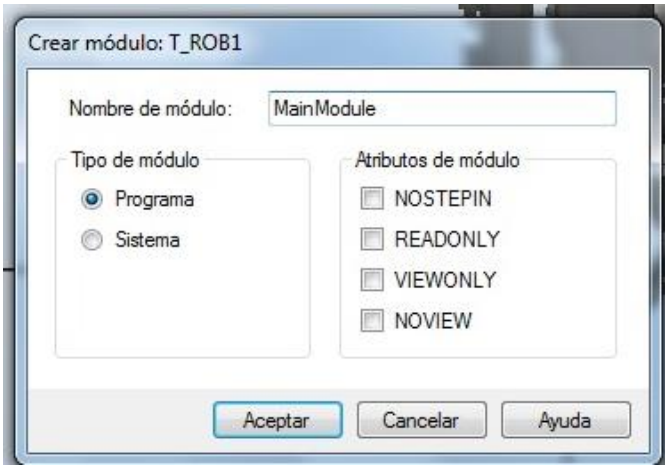


Ilustración 17. Nombrar y seleccionar tipo de módulo.

5. A continuación ya podemos escribir nuestro programa en el MainModule o podemos colocar más módulos a los que nuestro sistema realice las llamadas. Los módulos de Sistema “BASE” y “user” que estaban por defecto es recomendable eliminarlos.

6. Para guardar el programa que estamos haciendo, debemos hacer click sobre Programa y a continuación Guardar programa.

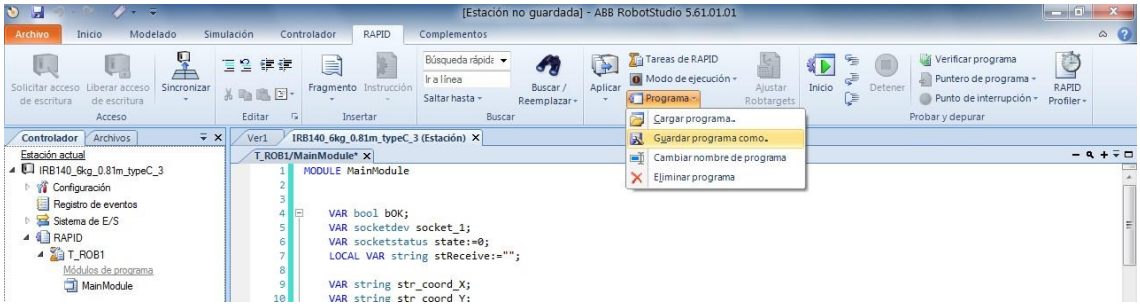


Ilustración 18. Guardando el programa.

7. Cuando guardamos el archivo, lo que hace el programa es crear una carpeta donde están dos elementos, el MainModule.mod y el programa de Rapid en PGF. Este último es el que usaremos en el robot.

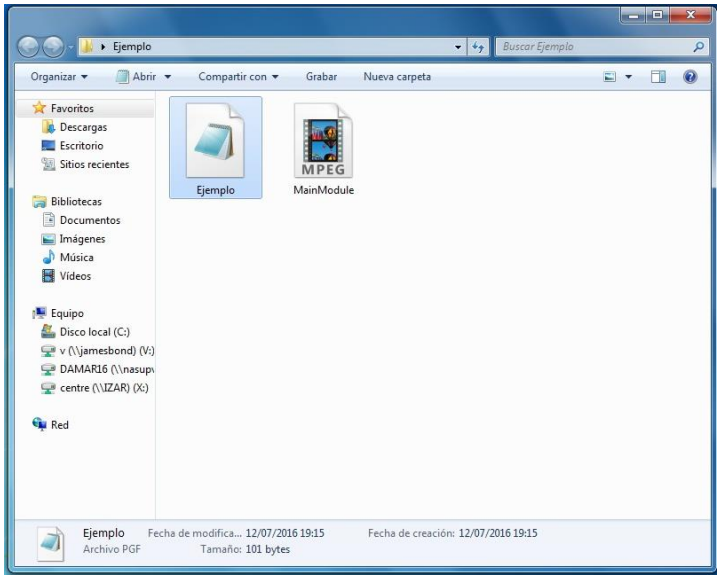


Ilustración 19. Programa guardado, archivo en PGF.

4.3 Conectarse al IPD

Dado que el IPD es el equipo que gestiona el sistema de visión artificial, debemos conectarnos a él mediante la aplicación de “Escritorio remoto”.

En nuestro PC, buscamos la aplicación “Conexión a escritorio remoto”, colocamos la IP del sistema, que es 158.42.16.207

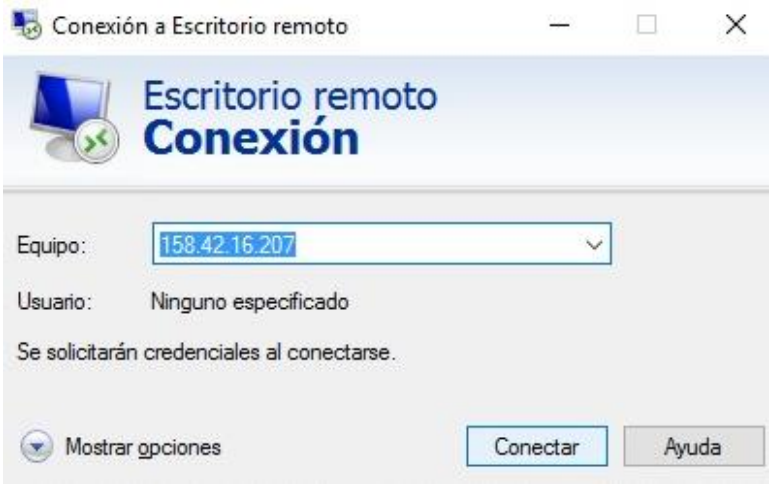


Ilustración 20. Conexión a escritorio remoto.

Tras pulsar en conectar, nos pedirá que pongamos un nombre de usuario y una contraseña, el cual nos habrá facilitado nuestro profesor o el personal del laboratorio.

Desde este momento ya podemos trabajar con el IPD.

5. Calibración

La calibración es un aspecto fundamental de este proyecto, ya que de ella depende que el objetivo se consiga o no.

Hay que calibrar dos elementos:

- Robot
- Cámara

5.1 Calibración del robot

Cuando calibramos un robot lo que hacemos es referenciar el eje o ejes a cero grados. Por eso es muy importante que todos los ejes que calibramos estén en la posición correcta, ya que un mal calibrado daría lugar a malos posicionamientos, a rotura de materiales y equipos o incluso daños personales. EL robot lleva unas marcas las cuales indican dónde se debe realizar la calibración. A continuación, explicaremos detalladamente cómo hacer la calibración paso por paso del robot:

1. En primer lugar colocaremos todos los ejes alineados. Para ello utilizaremos la función de movimiento desde el FlexPendant como vemos a continuación:
    a. Dentro del menú ABB pulsamos en “Movimiento”.

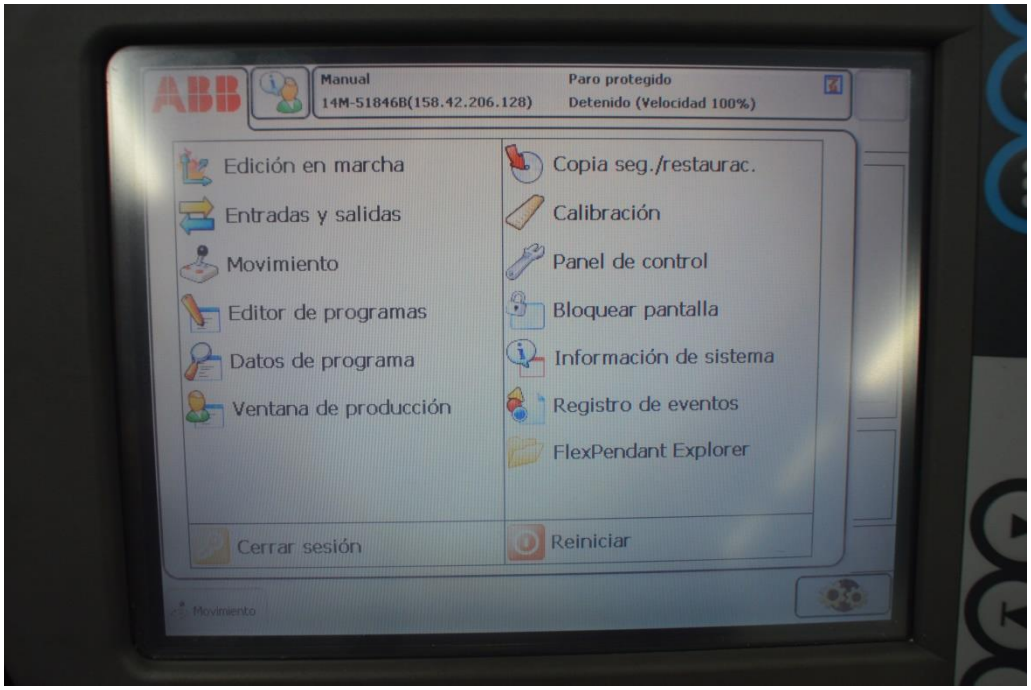


Ilustración 21. Selección Movimiento

b. En el menú “Movimiento” seleccionamos “Modo movto.”

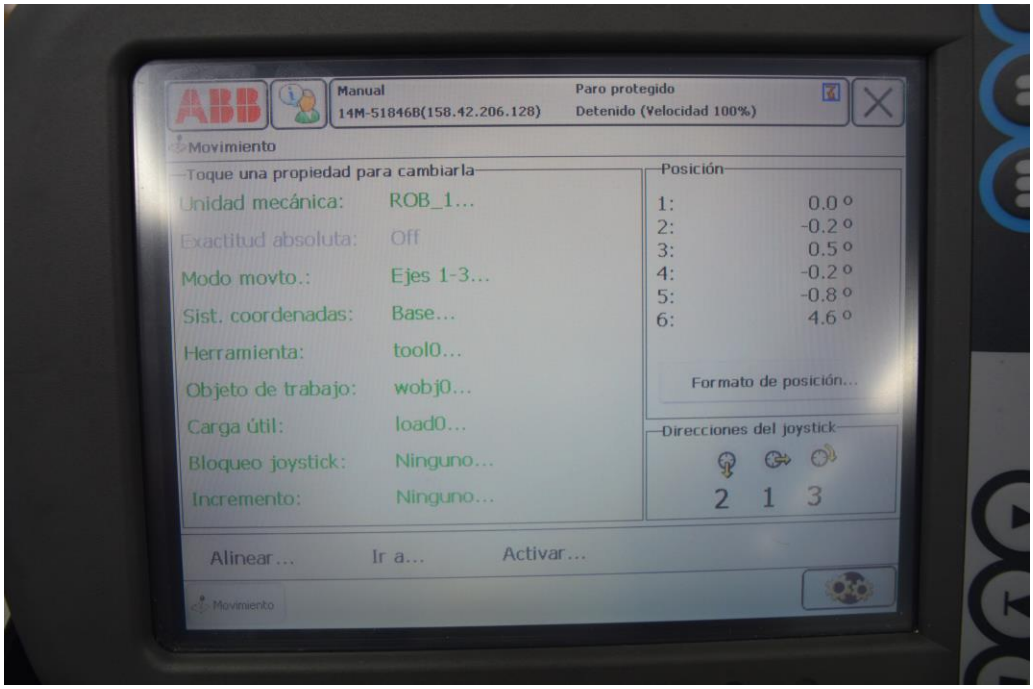


Ilustración 22. Selección Modo mvto.

c. Seleccionamos primero los ejes 1-3 o ejes 4-6 y pulsamos “OK”.

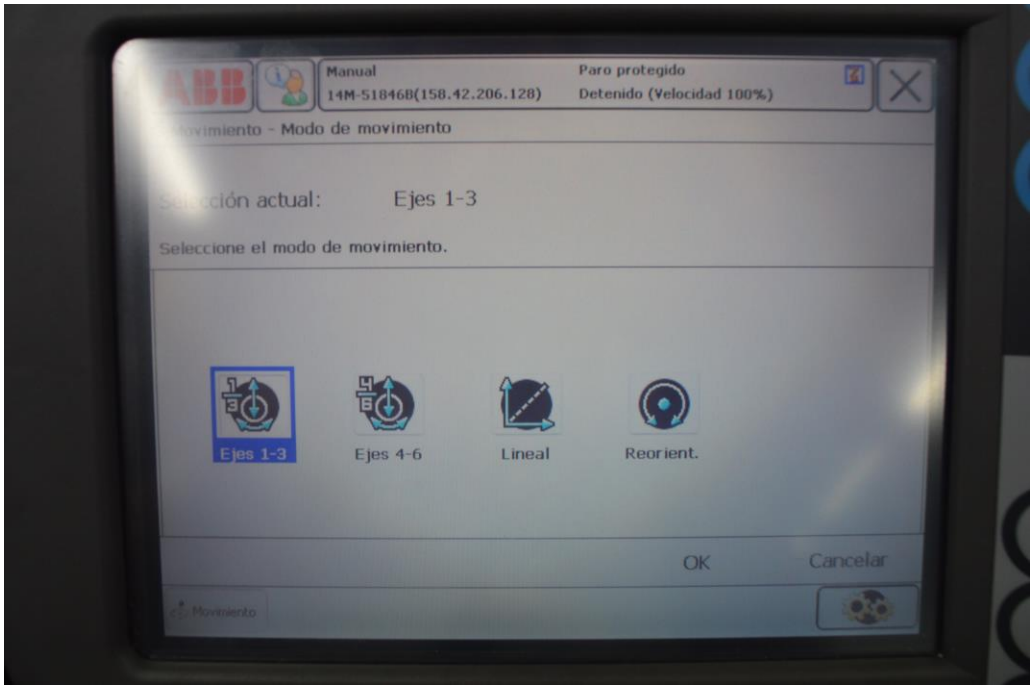


Ilustración 23. Selección de ejes.

Los ejes alineados deben quedar así:

Ilustración 24. Eje 1 alineado

Ilustración 25. Eje 2 alineado

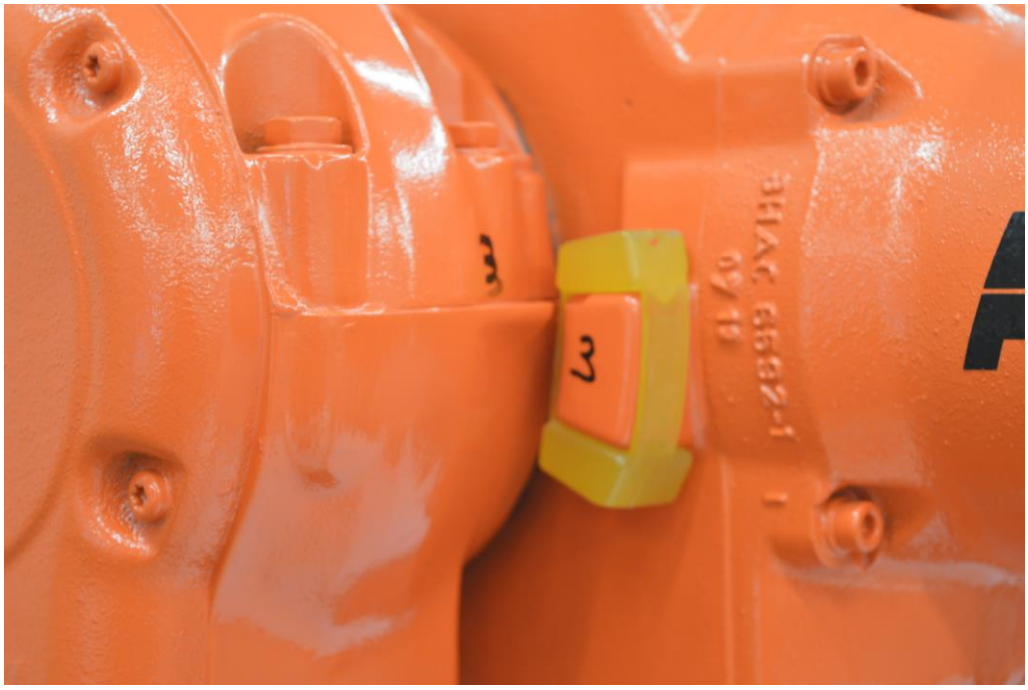


Ilustración 26. Eje 3 alineado

Ilustración 27. Eje 4 alineado

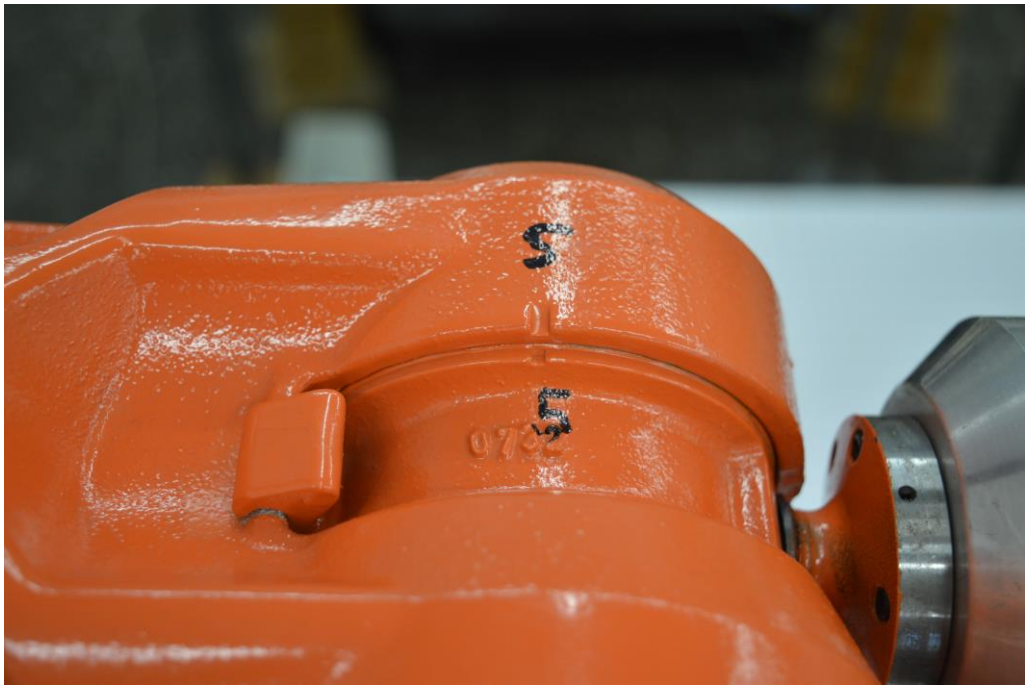


Ilustración 28. Eje 5 alineado

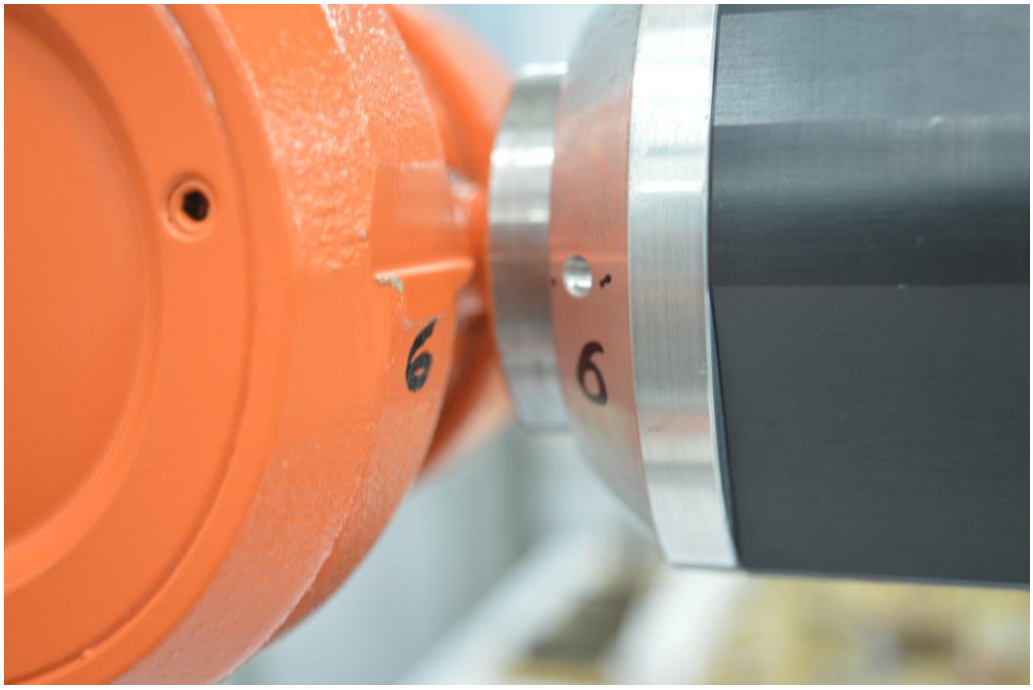


Ilustración 29. Eje 6 alineado

A continuación, para realizar el calibrado desde el FlexPendant realizaremos las siguientes instrucciones:

1. En el menú de “ABB” seleccionamos “Calibración”

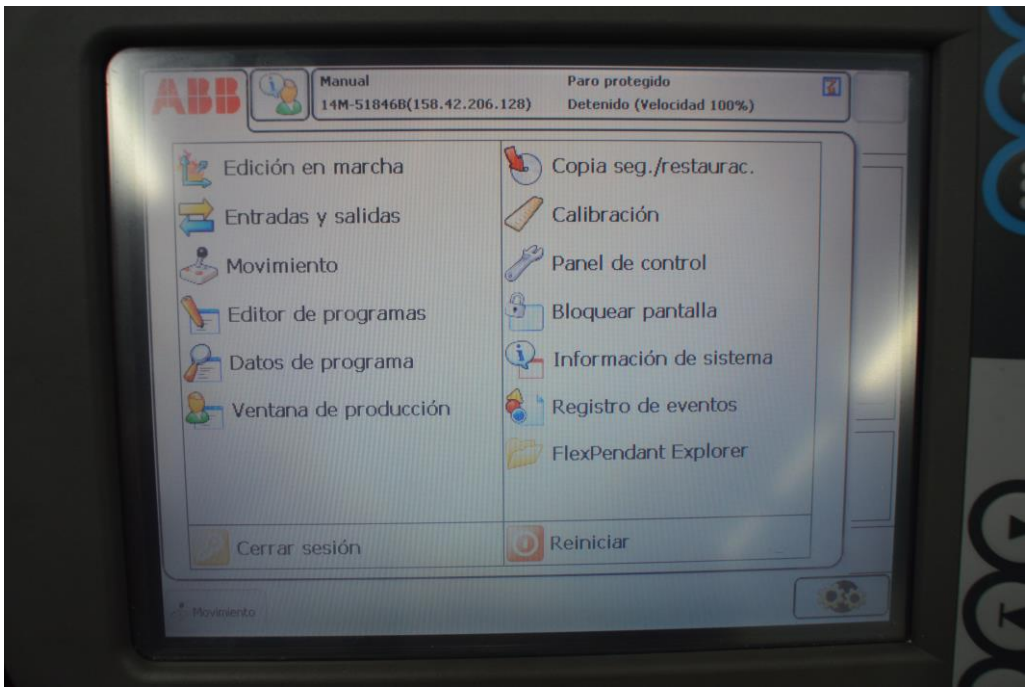


Ilustración 30. Selección Calibración.

2. Seleccionamos “ROB_1”

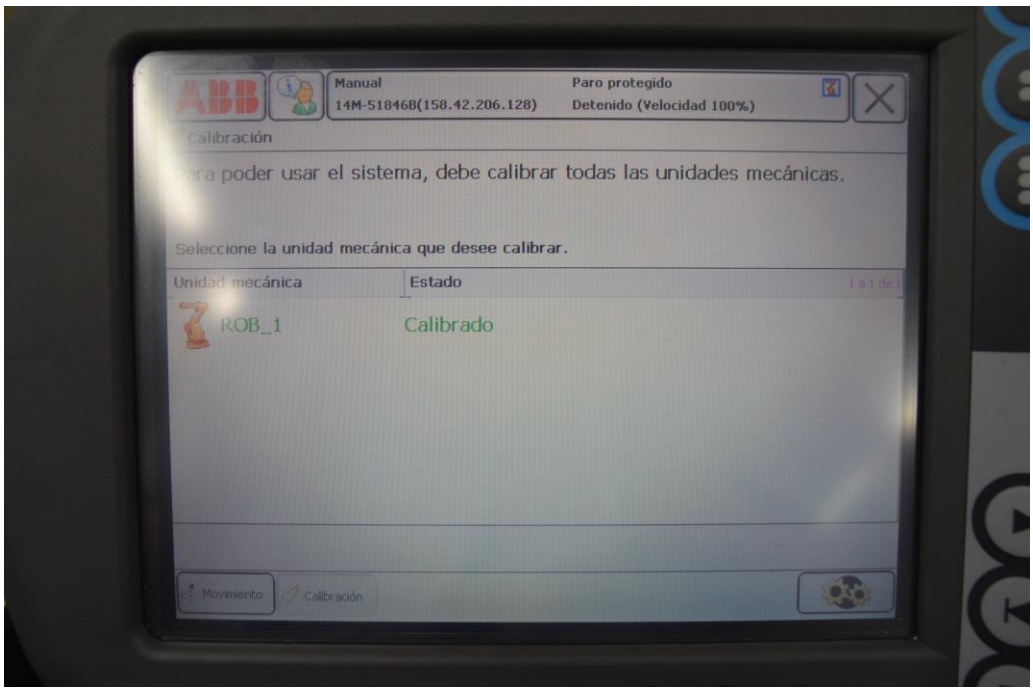


Ilustración 31. Selección ROB_1.

3. Pulsamos en “Actualizar cuentarrevoluciones”

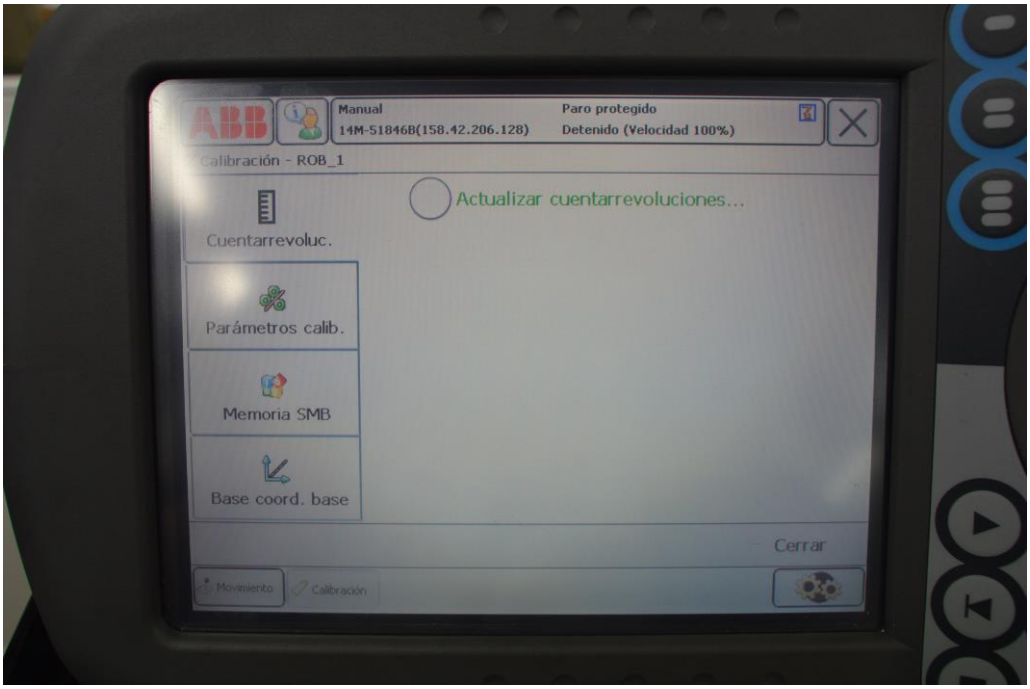


Ilustración 32. Actualizar cuentarrevoluciones.

4. Seleccionamos “Sí”

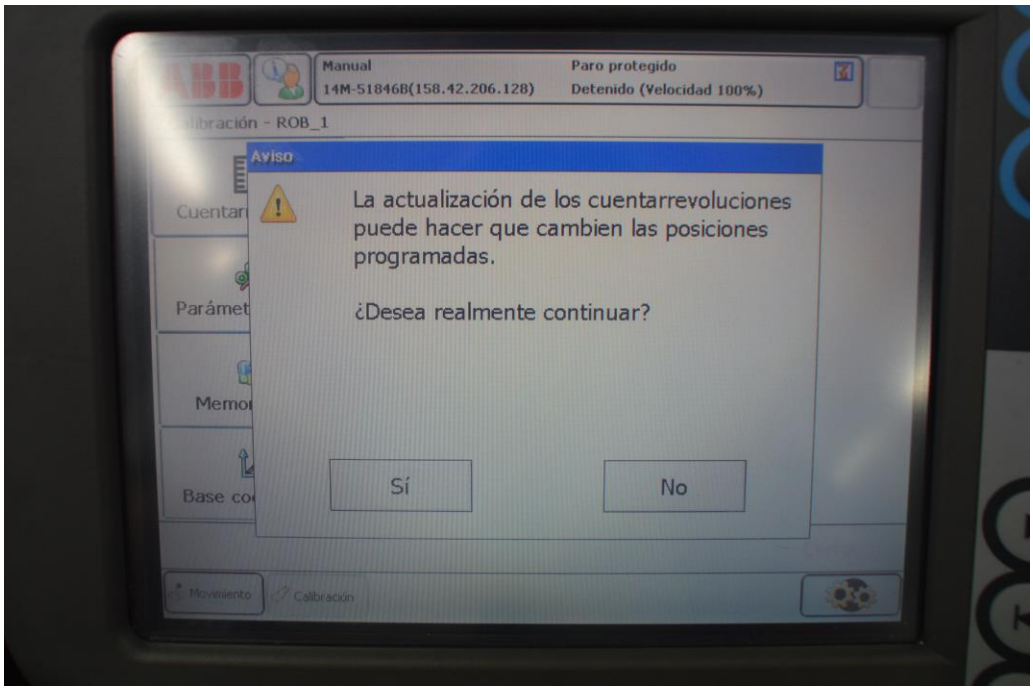


Ilustración 33. Confirmar cuentrarrevoluciones.

5. Seleccionamos todos los ejes

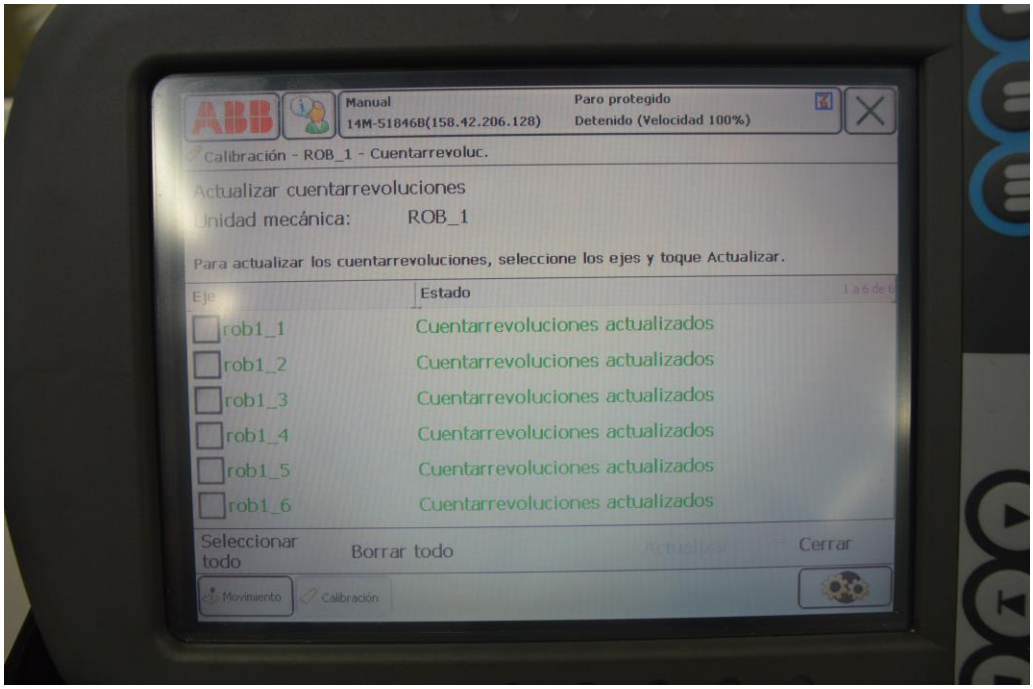


Ilustración 34. Selección de todos los ejes.

6. Pulsamos en “Actualizar”

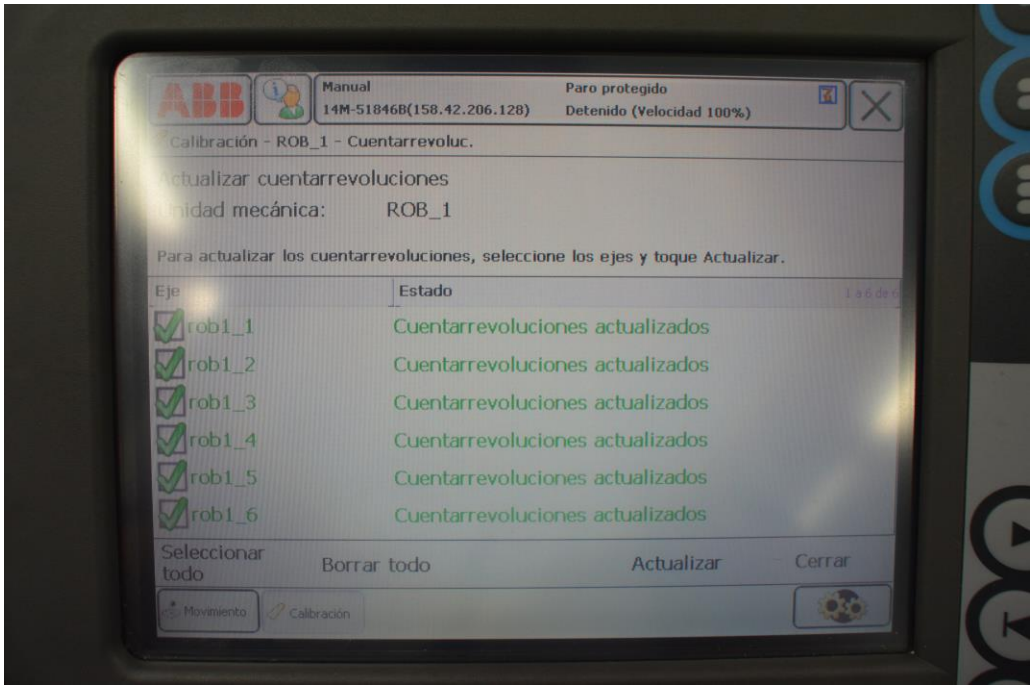


Ilustración 35. Pulsar Actualizar.

Desde ahora nuestro robot estará perfectamente calibrado.

5.2 Calibración de la cámara

La calibración de la cámara permite localizar los objetos de la escena con respecto al sistema de coordenadas de referencia (que en nuestro caso es el sistema de referencia base del robot).

Existen muchos métodos de calibración de cámaras. En las referencias [16-18] se describe el proceso de calibración completo de cámara y las condiciones óptimas de calibración.

5.2.1 Posibles soluciones

En primer lugar hemos de saber cómo matemáticamente se calibra la cámara para que nos dé un punto con las coordenadas del robot. En este caso, lo que debemos hacer es crear una ecuación que convierta las posiciones en pixeles que recibe la cámara (u,v) en posiciones que el estén en consonancia con el sistema de coordenadas del robot (x,y).

Hemos de aclarar que la coordenada Z será siempre la misma ya que, como solo tenemos una cámara, leemos en 2 dimensiones.

Dado que queremos convertir las coordenadas en píxeles (u, v) en coordenadas de robot (x, y), lo que debemos hacer es multiplicarlo por una matriz de transformación H, de esta forma:

$$(u, v) \cdot H = (x, y)$$

De manera que se nos presentan dos opciones para poder obtener la transformación:

- Calcular manualmente la matriz H.
  Pros:
  - No interfiere la altura.

  Contras:
  - Requiere más tiempo.
- Realizar la calibración desde el programa de visión artificial “Sherlock”.

  Pros:
  - Mayor rapidez.
  - Mayor capacidad de cambio.
  - Posibilidad de actualizar la calibración rápidamente.

Contras:

- Para realizar un sistema robusto hacen falta muchos puntos de calibración.
- Reducir los puntos de calibración puede dar lugar a errores si nos alejamos de los puntos donde hemos calibrado.
- No podemos calibrar dos zonas en diferentes alturas Z, a menos que utilicemos muchos puntos.

5.3 Solución escogida y justificación

Aunque podríamos calcular esta matriz a mano, es mejor que utilicemos las herramientas que nos proporcionan los propios programas de visión artificial. Por tanto, nos decidimos a hacer la calibración con el programa de visión artificial “Sherlock”.

Para realizarlo utilizaremos una tabla de referencias que existía ya en el laboratorio. Es la siguiente:

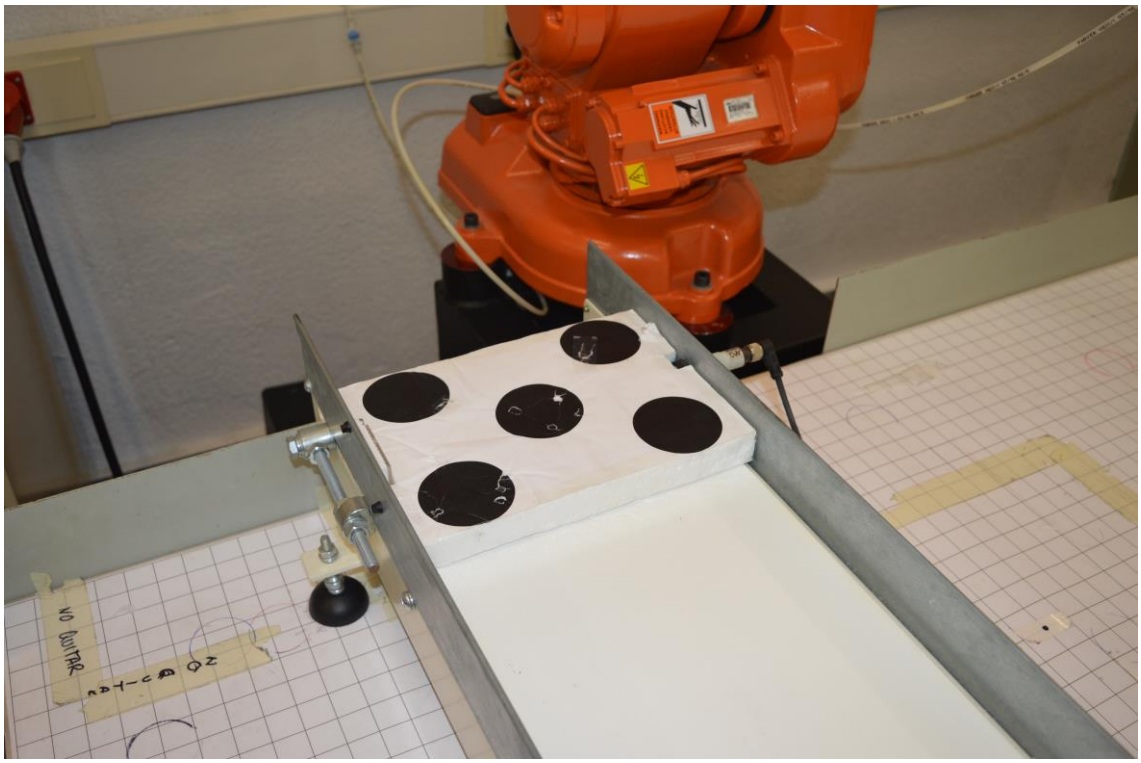


Ilustración 36. Patrón de calibración.

5.4 Procedimiento

Vamos a ver ahora por pasos cómo realizar esta calibración:

1. Moviendo el robot en manual, colocaremos el TCP (tool center point) del robot en uno de los círculos del patrón.

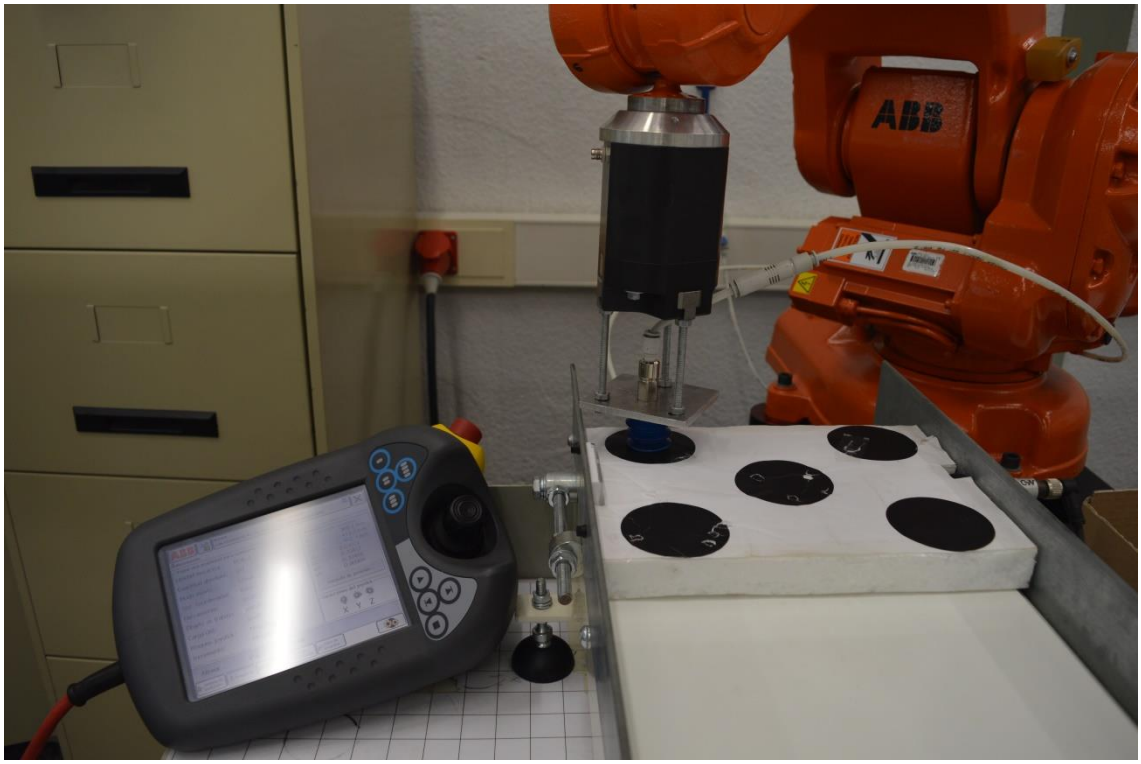


Ilustración 37. Mover el robot con el FlexPendant.

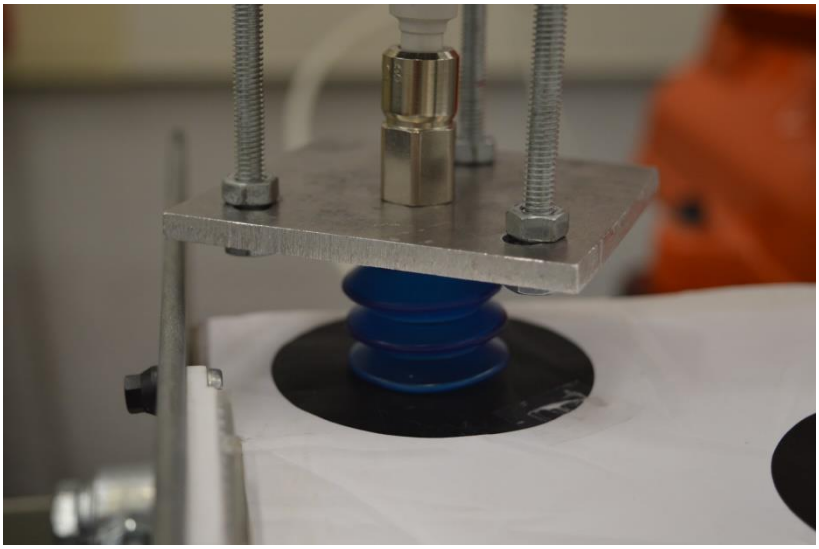


Ilustración 38. Detalle posición de calibración.

2. Anotamos la posición de X e Y que nos da el robot en la pantalla del FlexPendant. Volvemos a repetir los pasos 1 y 2 hasta tener los datos de todos los círculos.

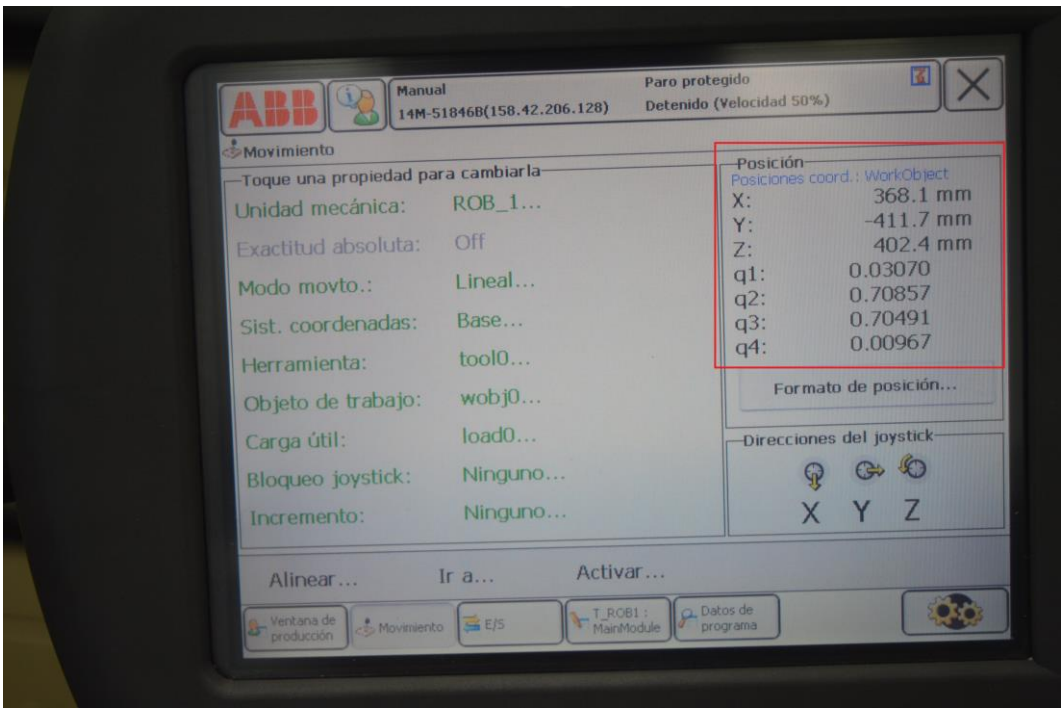


Ilustración 39. Anotar posiciones X e Y.

3. A continuación en Sherlock vamos a “Image Options” y después a “Calibration”. En la ventana añadimos una calibración y pulsamos “Calibrate using measurements”:

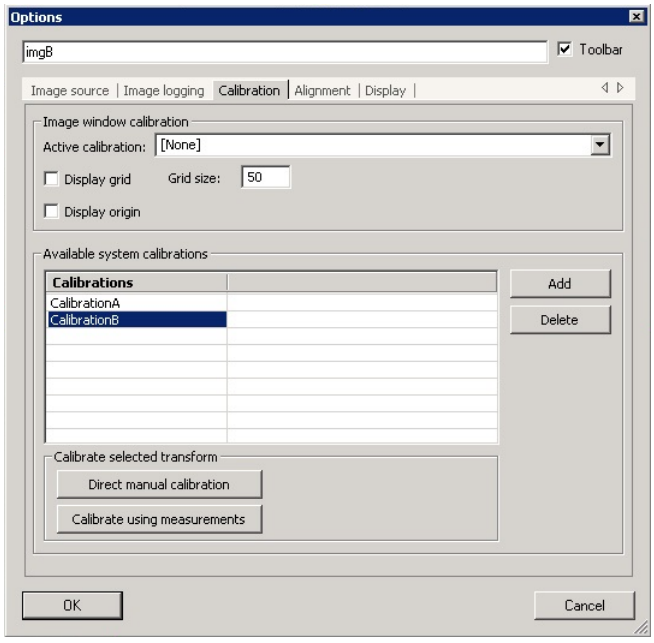


Ilustración 40. Seleccionar calibración.

4. En la siguiente ventana seleccionamos los centroides de los círculos de nuestro patrón, que habremos buscado anteriormente y clicando dos veces en cada centroide, indicaremos a qué medidas X e Y del robot corresponden. Cuando hayamos realizado al menos 5 puntos distintos, ya podremos pulsar en “Calibrate” para tener el sistema calibrado.
   Si nos fijamos, una vez calibrado, si pasamos el ratón por la pantalla, podremos observar cómo en la parte inferior izquierda de la ventana aparecen las coordenadas de Sherlock, en pixeles, y las correspondientes al robot, en X e Y.

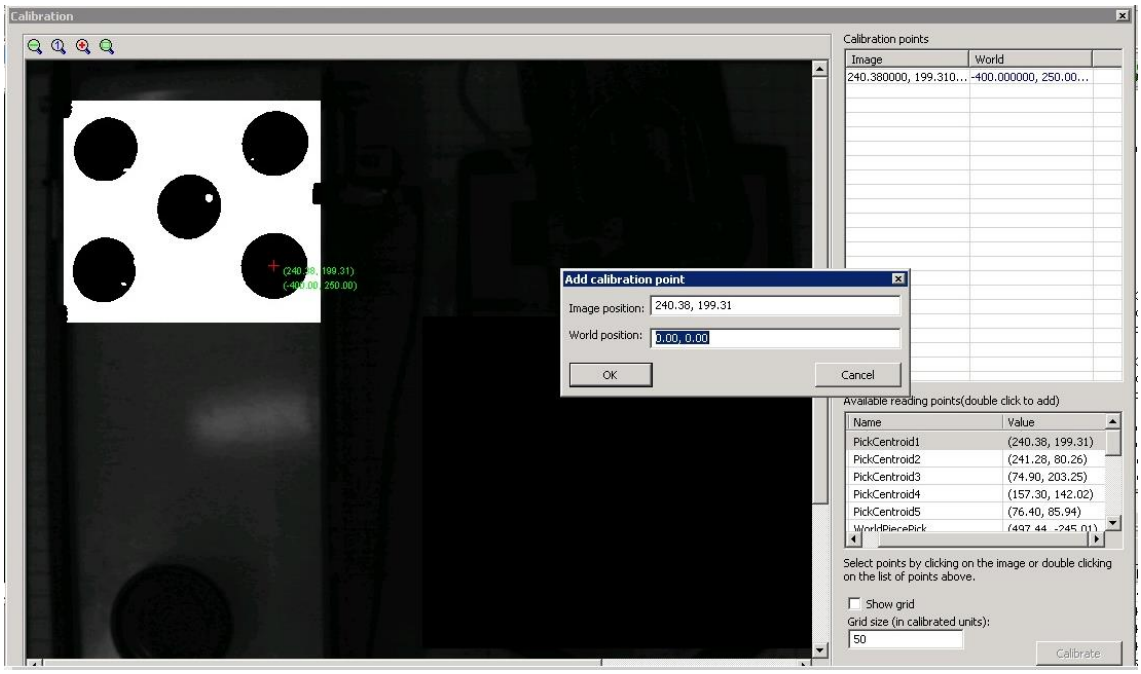


Ilustración 41. Asignar valores de X e Y a píxeles.

6. Solución para hallar las posiciones de las piezas y el envase.

6.1 Solución para las piezas

Dado que las piezas son conos con dos bases planas, debemos aprovechar las características de las mismas para hacer más fácil el trabajo de visión artificial.

Nos fijamos que las piezas en cuestión tienen en su base superior un color llamativo y fuerte, que puede contrastar muy bien con el fondo blanco de la cinta transportadora. Hemos de recalcar que si la pieza fuera en la posición inversa, la solución no podría ser la misma.

Por tanto la solución será:

1. Abrir el programa de Sherlock, con una pieza en la cinta y abrimos otra ventana donde ver la imagen en mono8.
2. Utilizamos la función rectángulo sobre la zona de la cinta donde el sensor detecta la pieza, es decir, la región de interés.

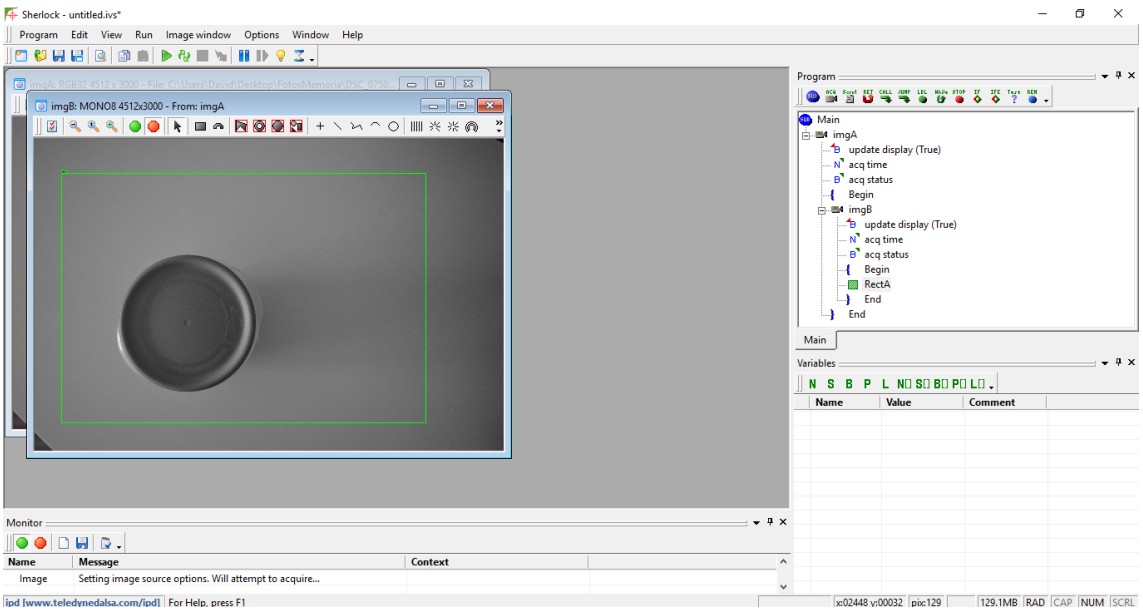


Ilustración 42. Colocación de la región de interés.

3. Dentro de la función cuadrado, seleccionamos Treshold y lo ajustamos de modo que sólo se vea la pieza.
4. Ajustamos el Connectivity Binary de modo que detecte la pieza, pero debemos ajustar los parámetros de altura, anchura y área de manera que, si por accidente entra otra pieza, no la reconozca. Debería quedar así:

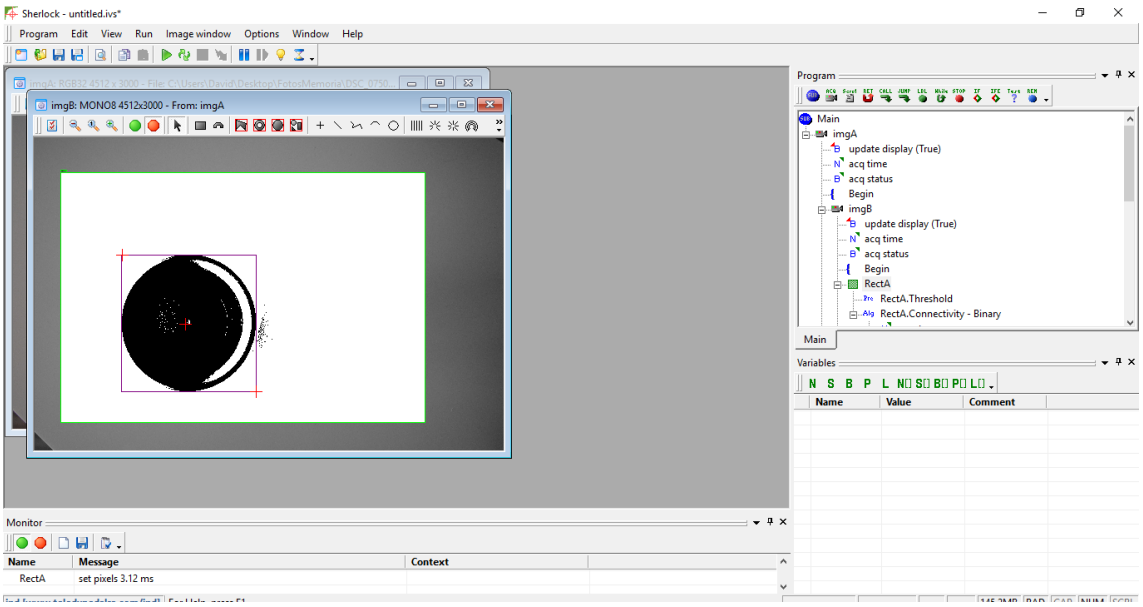


Ilustración 43. Selección de la pieza.

5. De este modo, en la variable Centroid del Connectivity Binary tendremos las coordenadas en pixeles de dónde estará la pieza.

6.2 Solución para el envase

Debido a que las paredes del envase no son lo suficientemente gruesas y, sobre todo, que todo el envase es del mismo color, lo cual dificulta diferenciarlo mediante visión artificial, vamos a instalar unos patrones en el envase de manera que serán ellos los que leeremos para saber dónde debemos colocar nuestra pieza.

Los patrones serán estos, hechos con cartulina negra:

Ilustración 44. Patrón para el envase.

Para no cometer errores a la hora de reconocer dónde debemos colocar las piezas, hemos de distinguir en el programa de visión artificial los patrones de las piezas ya colocadas.

Para eso seguiremos los siguientes pasos.

1. Crearemos otra imagen en Sherlock también con mono8.
2. Seleccionaremos con el rectángulo la zona de descarga, donde dejaremos las piezas en el envase. En la zona deben estar tanto una pieza como los patrones.
3. Realizaremos el Treshold como hicimos antes, de manera que los elementos queden bien distinguidos.
4. Con el Connectivity Binary modificaremos las mediciones de anchura, altura y área para que la pieza no sea seleccionada pero los patrones sí. Debe quedar de la siguiente manera:

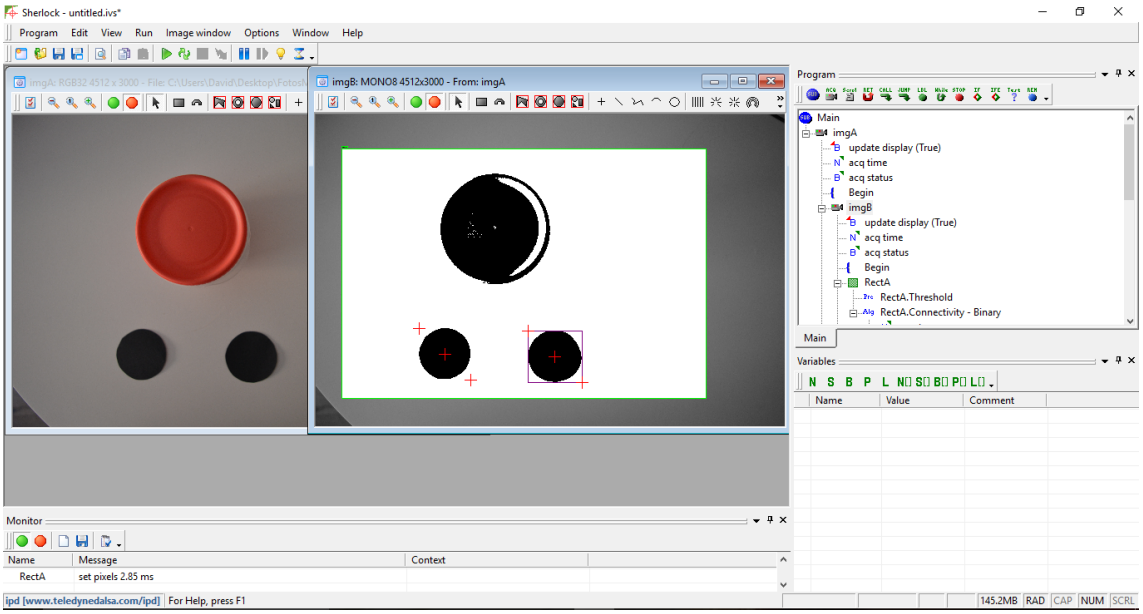


Ilustración 45. Diferenciación entre patrón y pieza.

Como observamos en la imagen anterior, la pieza no está seleccionada pero los dos patrones sí lo están. De esta manera, según se vayan colocando las piezas encima de los patrones, estas posiciones dejarán de estar disponibles porque las tapará la pieza cuyas dimensiones la cámara no acepta.

Por eso es tan sumamente importante hacer bien la calibración de la cámara, ya que de lo contrario, si el robot dejara la pieza separada del patrón y luego la cámara lo leyese, podríamos causar problemas, roturas y averías.

7. Comunicación robot – sistema de visión artificial

La comunicación entre el robot y el sistema de visión artificial se establece por medio de un socket y el uso de strings (o cadenas de caracteres). Se establece mediante la conexión IP con dirección 192.168.0.1.

7.1 Establecer conexión con el socket mediante rapid

Un ejemplo para establecer la conexión con el socket desde rápid sería esta:

```
PROC ConnectToSherlock()

SocketConnect socket_1, "158.42.16.207",1024;

ENDPROC
```

En este momento tendríamos creado el socket y podríamos utilizar ya la comunicación. Debemos poner la IP, que en este caso es 158.42.16.207 y el puerto, que en nuestro caso es 1024. Desde rapid podemos tanto enviar strings como recibirlas, de manera que podemos pedir particularmente datos y que el sistema de visión nos dé los que nosotros queremos.

7.2 Enviar datos (robot)

Para enviar strings utilizamos la siguiente instrucción:

```
SocketSend socket_1\Str:="STRING";
```

Siendo STRING la cadena de caracteres que deseamos enviar.

7.3 Recibir datos (robot)

Para recibir strings utilizamos la siguiente instrucción:

```
SocketReceive socket_1\Str:=stReceive;
```

La cadena de caracteres se almacenará en stReceive, que es el nombre que le hemos asignado como string, aunque puede ser cualquier otro.

7.4 Interpretar datos recibidos

Dado que los datos nos llegan desde la cámara mediante una string de tantos espacios como designemos en “Sherlock”, es importante que decidamos cómo seccionaremos los datos que nos van a llegar.

Según el manual que seguimos para hacer el proyecto, una manera era recibir tanto las coordenadas X e Y en una sola string. Tras recibirlas, debíamos seccionar la string para obtener X por un lado e Y por el otro. Se hace mediante los siguientes pasos:

```
SocketReceive ComSocket \Str:=stReceived;

XData:= StrPart(stReceived, 0, NumCharacters);
YData:= StrPart(stReceived, NumCharacters, NumCharacters);

bOK:=StrToVal(XData,nXOffs);
bOK:=StrToVal(YData,nYOffs);
```

Como observamos aquí arriba, mediante el socket obtenemos la cadena, con StrPart dividimos la cadena tantos espacios como deseemos, y con StrToVal lo que hacemos es convertir la cadena en una variable Val.

7.4.1 Problemática

A la hora de implementar este método tuvimos el siguiente inconveniente: como se realizan dos lecturas, una en la cinta y otra donde se coloca el envase, en una de ellas la Y era positiva y en otra negativa. Así mismo también, dependiendo la posición del envase, la Y podía ser negativa o positiva. De esta manera, no podíamos tener un seccionamiento fijo, ya que el signo negativo ocupa un espacio en la cadena y deberíamos recurrir a distintos seccionamientos dependiendo la posición, complicando más el programa.

7.4.2 Solución adoptada

Para solucionar este inconveniente, se ha adoptado la medida de solicitar individualmente al sistema de visión el punto X y el punto Y, de manera que no hay que seccionar la cadena que nos devuelve.

Pros:

- Evitamos los errores debido al cambio de signo de cualquier variable.
- Es más intuitivo y fácil a expensas de futuras actualizaciones y mantenimientos.

Contras:

- Más frecuencia de comunicación con el socket.

Puesto que son más las ventajas que los inconvenientes, y dado que en nuestro programa no tenemos problemas de tiempos, vamos a adoptar esta solución, que realizaremos de la siguiente manera:

```
SocketSend socket_1\Str:="VARX";

SocketReceive socket_1\Str:=stReceive;

bOK:=StrToVal(stReceive,XPoint);
```

Como vemos en el ejemplo, primero solicitamos a “Sherlock” la variable X, y éste nos la devuelve en cadena de caracteres, que nosotros debemos convertirá a Val mediante el comando StrToVal.

A continuación en los siguientes apartados, veremos cómo hacer la comunicación desde el programa Sherlock.

7.4 Configurar la conexión con el socket en Sherlock

La conexión se realiza mediante conexión TCP/IP. Vamos a configurar el sistema de visión como “servidor” y el robot será el cliente. Para realizarlo seguimos los siguientes pasos:

1. En Sherlock, vamos a Options → IO Configuration.
2. En la pestaña de TCP/IP, seleccionamos “Server” y asignamos el puerto 1024 (en caso de escoger otro puerto, deberemos cambiarlo también en la conexión del socket en el programa de Rapid.
3. Pulsamos añadir, y observaremos que se ha añadido un nuevo elemento que pondrá, en este caso: Server: 1024.
4. Aceptamos.

Debería quedar tal y como en la siguiente ilustración:

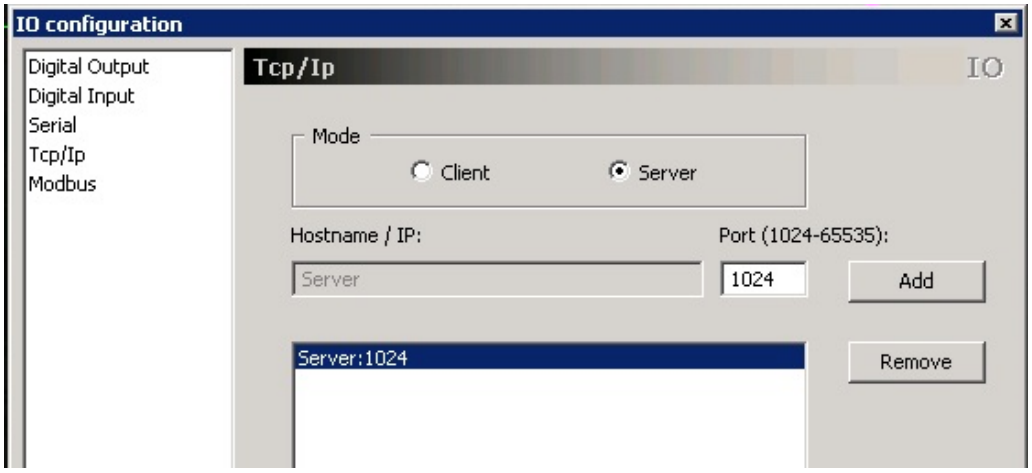


Ilustración 46. Configuración TCP/IP.

7.5 Enviar datos (Sherlock)

Para realizar la emisión de datos desde Sherlock utilizaremos la función "Send Line", que se puede encontrar en la librería de TCP/IP. Debemos tener claro que con esta función solo podemos enviar datos en forma de cadena de caracteres, o String. Es por eso que si estamos usando variables numéricas, booleanas o cualquier otras, debemos convertirlas a cadenas de caracteres. Otro detalle es que en la conexión debemos poner la que nosotros hemos creado, la 1024. La función quedará asi:

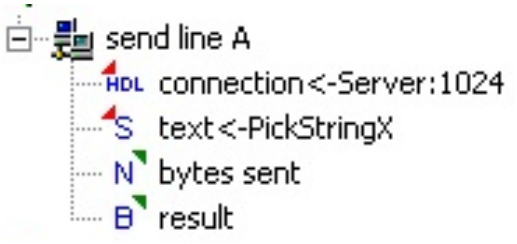


Ilustración 47. Send Line.

Como hemos dicho en connection pondríamos el puerto que hemos elegido y en text la cadena de caracteres.

7.6 Recibir datos (Sherlock)

Para recibir datos en Sherlock utilizamos la función "Receive Line", que se encuentra también en la librería de TCP/IP. Al igual que la función anterior, solo funciona con cadenas de caracteres y también debemos indicar la conexión que en este caso será la misma. Debería quedar así:

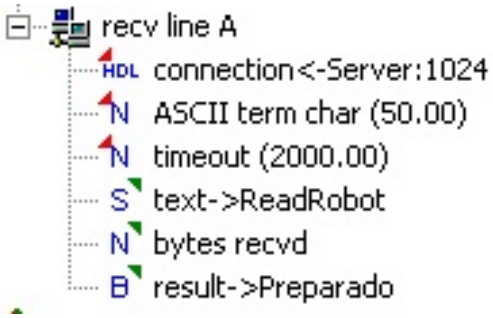


Ilustración 48. Receive Line.

Notamos como en esta función tenemos más opciones. Vamos a ver las que usaremos:

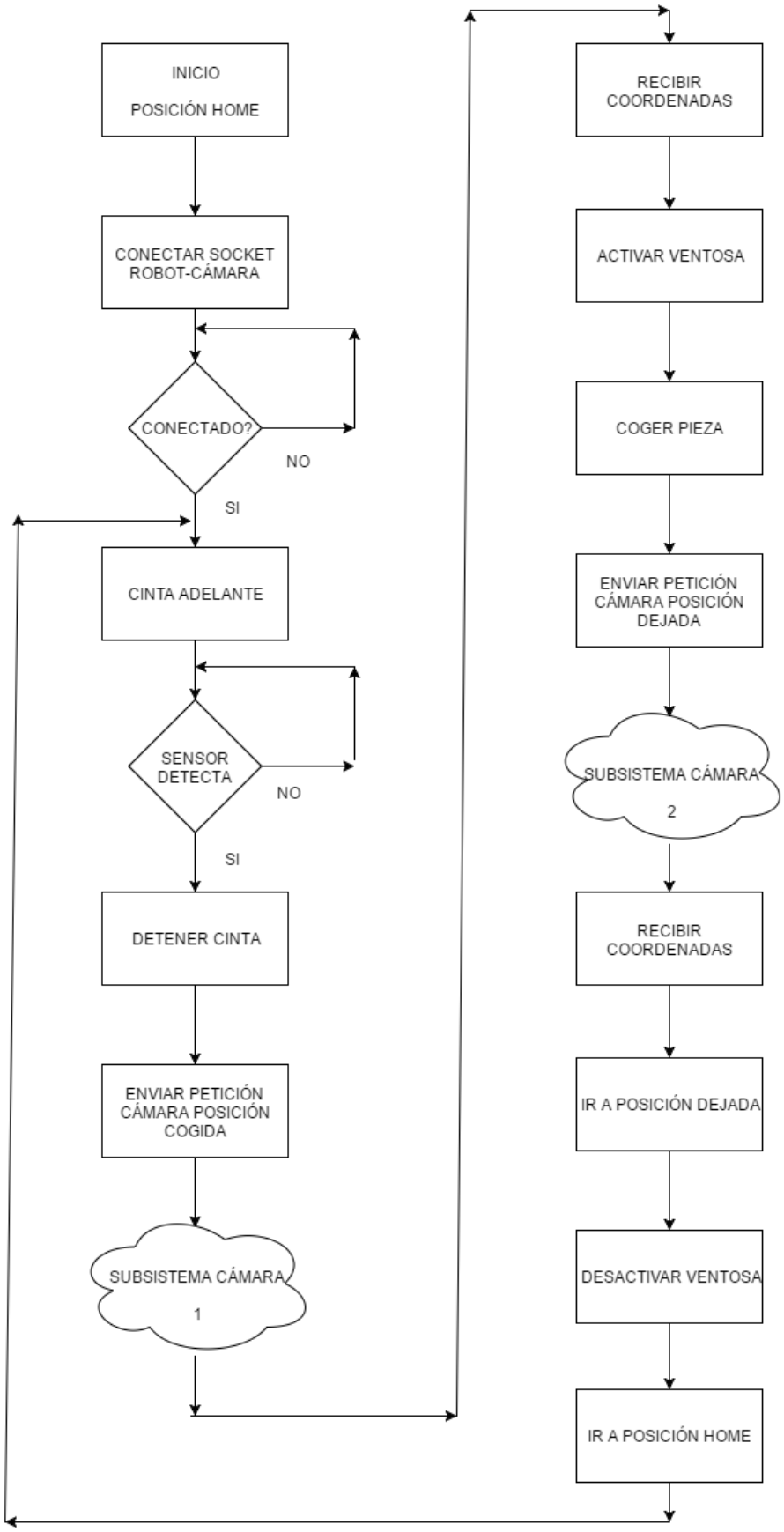
- Connection: debemos
- Timeout: nos indica cuánto tiempo espera para recibir el dato.
- Text: aquí se almacena la cadena de caracteres que nos envía el robot.
- Result: es un booleano, que pondrá a 1 la variable booleana que deseemos.

8. Diagramas de flujo

8.1 Diagrama de robot

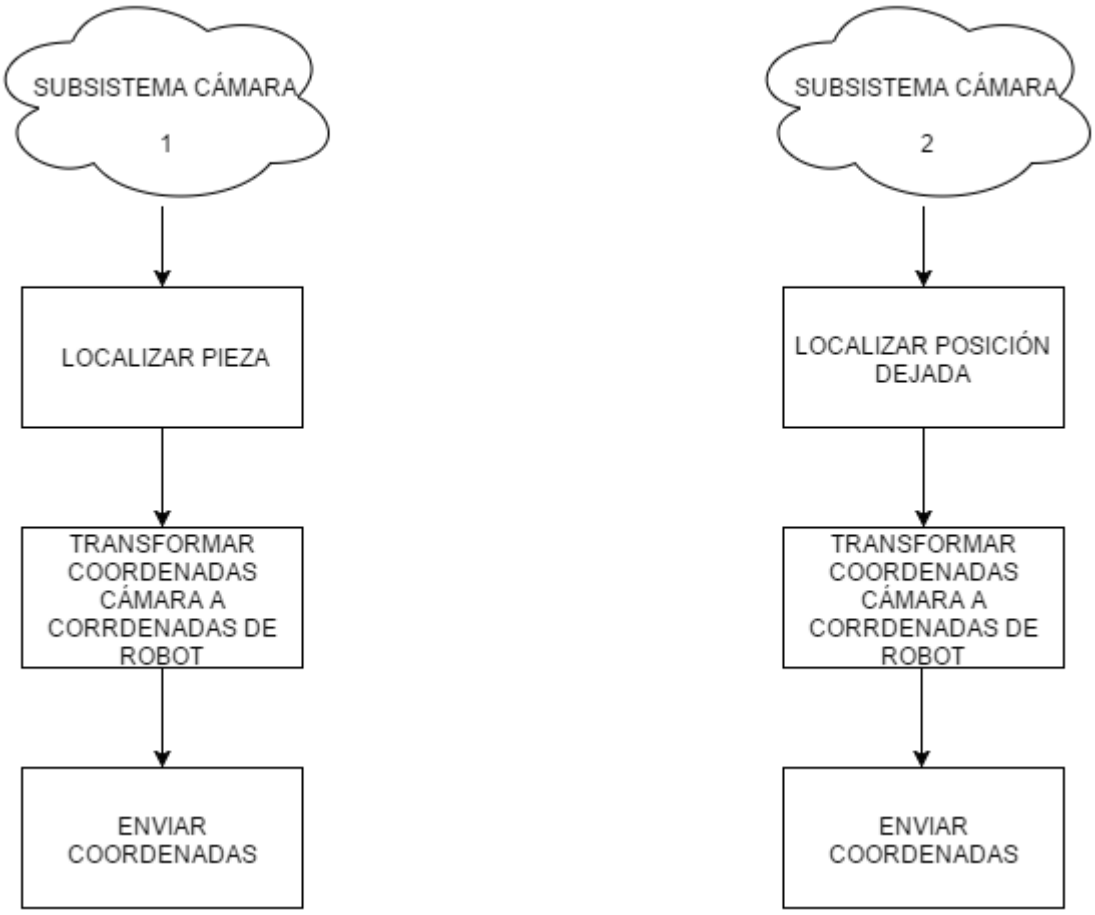
Los diagramas de flujo nos sirven para poder observar rápidamente cómo funciona un programa o sistema. No es necesario que se identifiquen todas las variables ni el programa detallado, sólo sirve para hacernos una idea rápida incluso para quien no tenga muchos conocimientos técnicos.

Particularmente para el programa de robot, este es el diagrama de flujo:



8.2 Diagrama de flujo del sistema de visión artificial

Al igual que en el diagrama de flujo del robot, en el diagrama de flujo de la cámara no se especifica con detalle el programa sino que se da una idea clara y rápida de cómo actúa el programa. Es el siguiente:



9. Descargar programa de Rapid en la Unidad de Control

Vamos a ver qué pasos detallados debemos seguir para poder utilizar el programa que hemos realizado en RobotStudio en el robot:

1. Para descargar nuestro programa en el robot lo más útil es utilizar una memoria USB que conectaremos al puerto USB que tiene esta Unidad de Control, tal como se aprecia en la siguiente ilustración.

Ilustración 49. Puerto USB de la Unidad de Control.

2. En el FlexPendant, pulsamos el menú de ABB y a continuación pulsamos en Ventana de Producción. En ella pulsamos en “Cargar programa…”

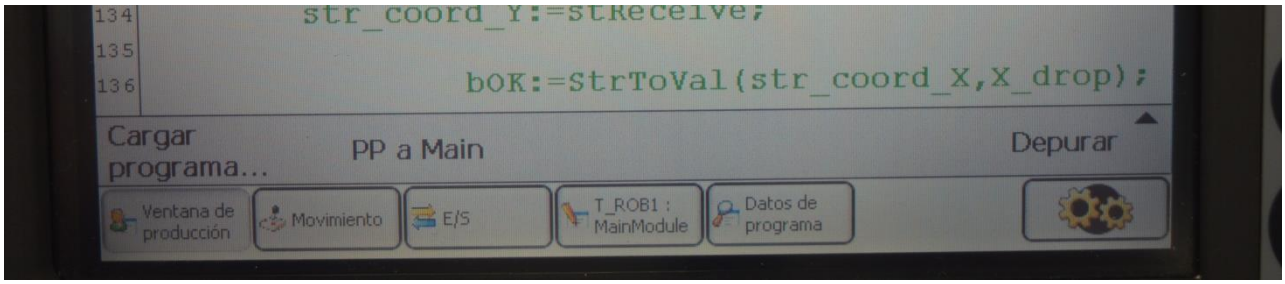


Ilustración 50. Cargar programa.

3. En caso de que tuviese algún programa ya cargado nos aparecería una ventana que nos avisa si queremos continuar ya que los datos del antiguo programa se perderán. Seleccionamos “Si”.

4. En la ventana siguiente podremos explorar tanto el disco duro de la Unidad de Control como nuestra memoria USB. Debemos abrir las carpetas hasta llegar a nuestro programa y pulsar en “OK”.

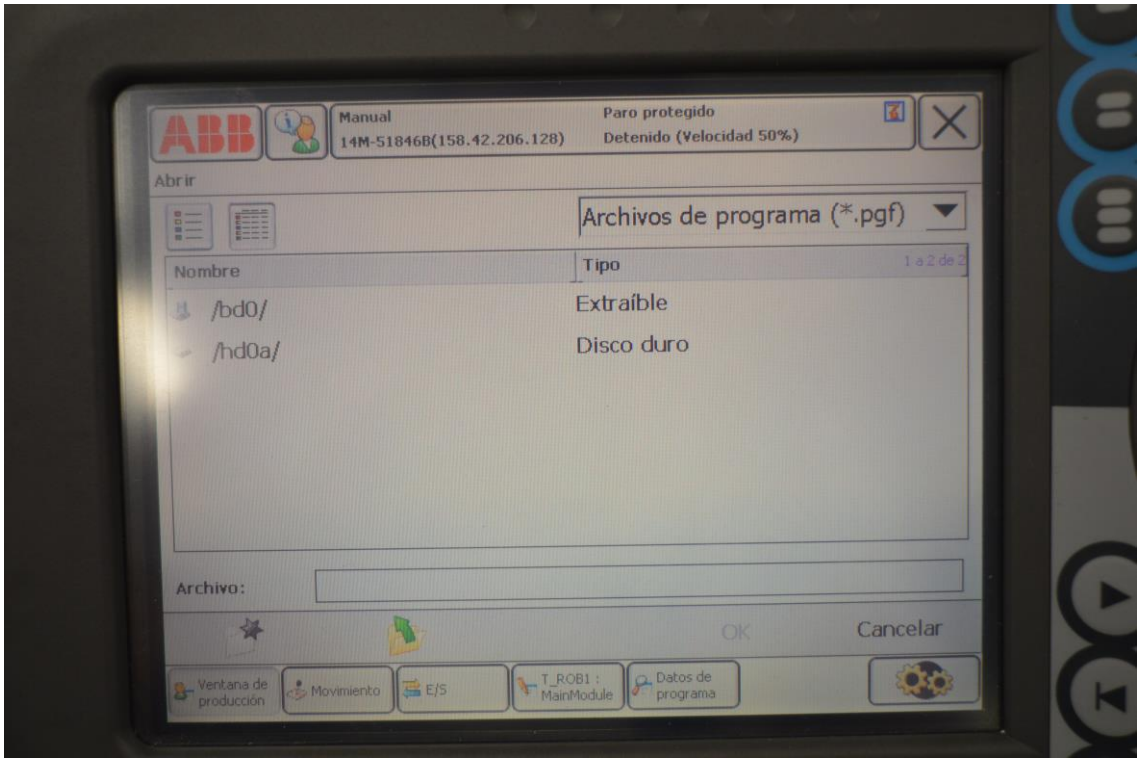


Ilustración 51. Buscar programa y cargarlo.

10. Arranque del proceso de inspección y manipulación de piezas

A continuación vamos a ver el proceso mediante el cual funcionarán conjuntamente tanto el sistema de visión artificial como el sistema de robot.

Dado que, como recordamos, hemos configurado a Sherlock como Servidor, éste deberá arrancarse primero, ya que de lo contrario obtendríamos errores.

Por eso, paso por paso, la secuencia de inicio de producción teniendo en cuenta que los programas ya están completados y cargados tanto en el IPD como en la Unidad de Control del robot es:

1. Abrimos Sherlock, en la barra de herramientas pulsamos sobre "Program" → "Open" y buscamos la carpeta donde está nuestro proyecto. Otro método es abrir directamente nuestro programa de Sherlock haciendo doble click sobre él.
2. Arrancamos Sherlock. Existen dos modos, vemos el Play en verde que solo hace un ciclo, y el de su derecha hace una marcha contínua. Debemos elegir este último.

Ilustración 52. Selección modo continuo.

Seguidamente, arrancaremos el robot. Damos por hecho que el robot ya está encendido y tiene el programa de Rapid cargado. Los pasos son:

1. Giramos la llave de la Unidad de Control hacia la izquierda, pasando de Manual a Automático.
2. Pulsamos el botón blanco, que sirve para activar los motores.

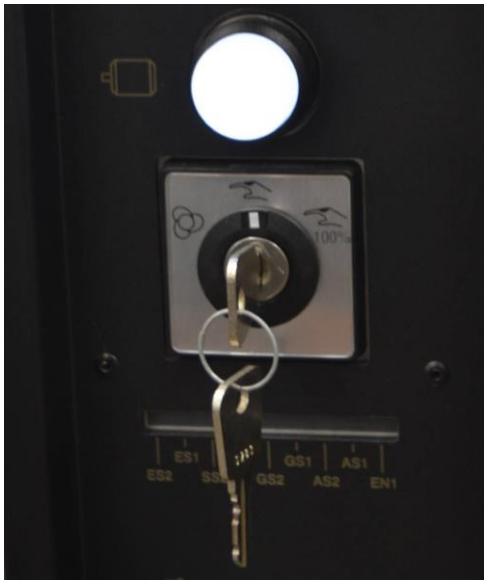


Ilustración 53. Selección de modo.

3. Finalmente en el FlexPendant pulsamos el botón de Play.

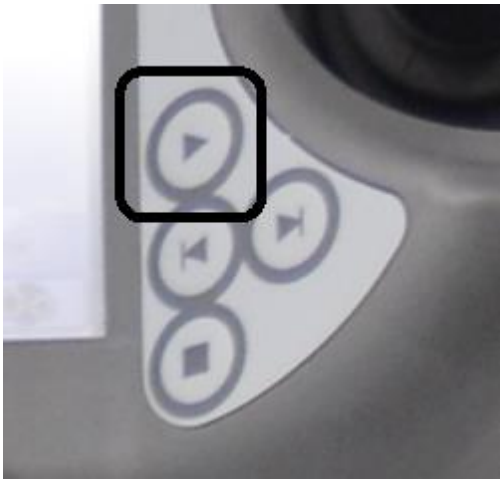


Ilustración 54. Botón Play.

10.1 Detener el proceso

Si deseamos detener el proceso debemos parar primero el robot y después el sistema de visión artificial, ya que de lo contrario podría dar problemas de comunicación.

Para detener el robot debemos pulsar sobre la tecla de Stop situada en el FlexPendant:

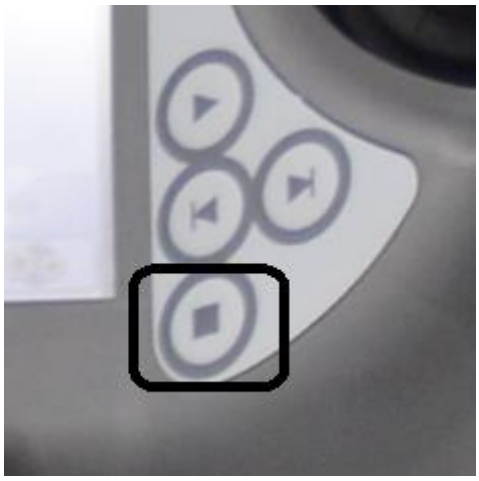


Ilustración 55. Botón Stop.

Para detener el programa de Sherlock, debemos pulsar en el siguiente botón:

Ilustración 56. Botón finalizar proceso Sherlock.

A continuación deberemos llevar el robot a home y resetear los valores de Sherlock. O bien si no queremos resetear todos los valores de Sherlock, podemos cerrar y abrir el programa.

También debemos tener en cuenta que debemos de llevar el puntero de programa del robot a Main antes de empezar de nuevo.

11. Dificultades y problemas encontrados

Durante el transcurso del trabajo de fin de grado surgieron varios problemas, dos de ellos importantes que costaron su tiempo en solucionar y que deben quedar reflejados en un apartado y también sobre cómo se ha solucionado.

El primero de ellos es el problema para cargar el programa de Rapid en la Unidad de Control. Tras hacer el programa entero en RobotStudio al pasar el PGF a la Unidad de Control esta no lo reconocía.

El problema se debía a que estábamos programando dentro del módulo del sistema y no dentro del módulo de programa. Por ello, se ha detallado cuidadosamente en el apartado “4.2 Crear programa en RobotStudio” cómo crear un programa de Rapid que no dé problemas en la Unidad de Control.

El segundo problema que se presentó fue con el troceamiento de la cadena de caracteres que recibíamos de Sherlock. Como dependiendo de la posición del envase la posición Y podía ser positiva o negativa, esta añadía o quitaba un signo negativo, lo cual impedía trocear esta señal siempre igual, ya que perdíamos información o, mejor dicho, era errónea.

En el apartado “6.4 Interpretación de datos recibidos” se explica qué medida se utilizó para atajar este inconveniente.

12. Valoración presupuestaria del proyecto y amortización

En primer lugar valoramos lo que nos cuesta mantener a los empleados trabajando. Después lo compararemos con el presupuesto que nos ha facilitado (y que se encuentra en un documento adjuntado) y observaremos si merece la pena o no.

12.1 Gastos actuales

Comprobaremos cuánto nos cuesta ahora al año los tres operarios trabajando en turnos de 8 horas 3 turnos al día de lunes a viernes: mañana, tarde y noche.

Operarios	Salario(€) bruto anual	Coste anual SS (€)	Total €/año
OP 1	19.600,00	5.880,00	25.480,00
OP 2	19.600,00	5.880,00	25.480,00
OP 3	19.600,00	5.880,00	25.480,00
Total	58.800,00	17.640,00	76.440,00

Tabla 1. Salarios operarios.

De este modo podemos observar que, anualmente, utilizamos en sueldos 76.440,00 €, teniendo en cuenta que es posible que cojan bajas, vacaciones, etc.

12.2 Presupuesto de automatizar la línea

El total el presupuesto de automatizar la línea que recoge el documento de “Presupuesto” es de **102.165,22 €.**

12.3 Conclusiones

Si comparamos lo que nos cuesta mantener un año los salarios de los operarios con el presupuesto de automatizar la línea observamos que se puede amortizar el desembolso en un año y tres meses. De hecho en un año habríamos amortizado el 75% de la inversión.

También debemos contar con que los robots y los sistemas no cogen bajas, apenas necesitan mantenimiento, no depende su rendimiento del estado de ánimo o de la edad, etc.

Por tanto la conclusión que se recoge es que automatizar las líneas de producción supone un ahorro muy considerable a medio plazo, rentabilizándolo al máximo con el paso de los años, haciendo de nuestra empresa una empresa más flexible y competitiva, pudiendo desarrollar costes de producción bajos sin tener que fabricar/manipular en países con mano de obra barata.

De este modo la automatización se erige como parte fundamental del desarrollo competitivo en los países industrializados.

13. Normativa de seguridad

13.1 Normativa estandarizada

La normativa respectiva al sistema de visión artificial en cuanto a medidas de seguridad y salud son las reflejadas en el Real Decreto 1215/1997.

La normativa respectiva vigente para los robots industriales:

- UNE-EN ISO 10218-1:2011 Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots.

- UNE-EN ISO 10218-2:2011 Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas robot e integración.

- ISO/TR 13309:1995 Manipulación de robots industriales. Guía informativa sobre los equipos de ensayo y métodos de metrología de operación para la evaluación del desempeño del robot de acuerdo con la norma ISO 9283.

- ISO 9283:1998 Robots manipuladores industriales. Criterios de análisis de prestaciones y métodos de ensayo relacionados.

- ISO 9409-1:2004 Manipulación de robots industriales. Interfaces mecánicas. Parte 1: Placas.

- ISO 9409-2:2002 Manipulación de robots industriales. Interfaces mecánicas. Parte 2: Ejes.

- ISO 9787:2013 Robots y dispositivos robóticos. Sistemas de coordenadas y nomenclaturas de movimiento.

- ISO 13482:2014 Robots y dispositivos robóticos. Requisitos de seguridad para robots para el cuidado personal.

- ISO 8373:2012 Robots y dispositivos robóticos. Vocabulario.

- ISO 14539:2000 Robots manipuladores industriales. Transporte de objetos con dispositivos de agarre tipo empuñadura. Vocabulario y presentación de características.

13.2 Normas de seguridad en el laboratorio

- ✓ Está prohibido el acceso al recinto con el robot en funcionamiento.
- ✓ Los permisos de trabajo deben especificar claramente los siguientes puntos:
  - Trabajos a realizar.
  - Personal que va a realizar el trabajo.
  - Especificar medidas de seguridad a adoptar.
  - Especificar pasos a seguir.
- ✓ Eliminar la alimentación del robot y de los demás equipos imposibilitando sus movimientos.
- ✓ Condenar botonera o dispositivos de parada.
- ✓ En el caso de tener que trabajar con el robot alimentado, se deberá hacer en manual a velocidad reducida.
- ✓ Si es posible, la programación deberá realizarse fuera de la zona de trabajo del robot.
- ✓ Durante la programación y pruebas sólo se permitirán velocidades bajas.
- ✓ La programación deberá efectuarse sólo por personal cualificado y autorizado.

14. Anejo

14.1 Ilustraciones

14.2 Tablas

15. Bibliografía utilizada

[1] http://www.profesormolina.com.ar/tecnologia/robotica/historia.htm

[2] http://robotiica.blogspot.com.es/2007/10/historia-de-la-robtica.html

[3] Valera Fernández, Á., Gómez Moreno, J., Sánchez Salmerón, A. J., Ricolfe Viala, C., Zotovic Stanisic, R., & Vallés Miquel, M. (2012, October). Industrial robot programming and upnp services orchestration for the automation of factories. In International Journal of Advanced Robotic Systems (Vol. 9, pp. 1-11). InTech.

[4] Valera, A., Vallés, M., Díez, J. L., Pizá, R., & Sánchez, A. Utilización de Sensorización Externa en Robótica. XXIV Jornadas de Automática, España, ISBN, 84-931846.

[5]http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/historia.html

[6] "Conceptos y métodos en Visión por Computador". Varios autores. Editado por: Alegre Gutiérrez, Enrique; Pajares Martinsanz, Gonzalo; de la Escalera Hueso, Arturo. ISBN: 978-84-608-8933-5, Junio 2016.

[7] Benlloch, J. V., Agusti, M., Sanchez, A., & Rodas, A. (1995, October). Colour segmentation techniques for detecting weed patches in cereal crops. In Proc. of Fourth Workshop on Robotics in Agriculture and the Food-Industry (pp. 30-31).

[8] Benlloch, J. V., Sanchez, A., Christensen, S., & Walger, M. (1996). Weed mapping in cereal crops using image analysis techniques. AgEng96, 2, 1059-1060.

[9] Benlloch, J. V., Sánchez, A., Agustí, M., & Albertos, P. (1996). Weed Detection in Cereal Fields Using Image Processing Techniques. Precision Agriculture, (precisionagricu3), 903-903.

[10] Ivorra, E., Sánchez, A. J., Verdú, S., Barat, J. M., & Grau, R. (2016). Shelf life prediction of expired vacuum-packed chilled smoked salmon based on a KNN tissue segmentation method using hyperspectral images. Journal of Food Engineering, 178, 110-116.

[11] ABAD RUIZ, SERGIO (2015). Reconocimiento de características de interés en una cadena de envasado de naranjas mediante visión artificial (Doctoral dissertation).

[12] Sánchez, A. J., & Martínez, J. M. (2000). Robot-arm pick and place behavior programming system using visual perception. In Pattern Recognition, 2000. Proceedings. 15th International Conference on (Vol. 4, pp. 507-510). IEEE.

[13] Pico, J., Sanchez, A., Bondia, J., Escudero, D., Torregrosa, A., & Correcher, C. (2005). Intelligent robotic cell for Trencadis mosaics manufacturing. IEEE Transactions on Systems Man and Cybernetics Part C(Applications and Reviews), 35(1), 75-86.

[14] VICENT FERRANDO, LLUÍS (2015). Automatización del sistema de envasado de producto mediante robótica y visión.(envasado automático de cartuchos de tinta) (Doctoral dissertation).

[15] ROCA ALCARAZ, JV (2015). Guiado de un robot ABB mediante sistema de visión (Doctoral dissertation).

[16] Viala, C. R., & Salmerón, A. J. S. (2008). Procedimiento completo para el calibrado de cámaras utilizando una plantilla plana. Revista Iberoamericana de Automática e Informática Industrial RIAI, 5(1), 93-101.

[17] Viala, C. R., Salmerón, A. J. S., & Fernández, R. S. (2003). Técnicas de calibrado de cámaras. Journal de Mathematiques Pures et Appliques, Leon.

[18] Ricolfe-Viala, C., & Sanchez-Salmeron, A. J. (2011, September). Optimal conditions for camera calibration using a planar template. In 2011 18th IEEE International Conference on Image Processing (pp. 853-856). IEEE.